# AAMC-PCI-MP

# 用户手册

V1.0

北 京 神 州 航 宇 测 控 技 术 有 限 公 司

# 目 录

# 手册内容简介

本手册内容共分为四个章节，分别为概述、硬件使用说明、驱动接口说明和应用程序说明。

第一章—概述，是对硬件功能与使用方法的整体概述，更多的细节在第二章—硬件使用说明、第三章—驱动接口说明和第四章—应用程序说明中进行详细描述。

第二章—硬件使用说明，描述了硬件的使用环境、配置安装方法、硬件结构及用户在实际使用中可能用到的硬件接口。

第三章—驱动接口说明，该部分内容是本手册的重点内容，它详细描述了接口函数的功能及使用方法，通过这一章节的了解，用户可以进行产品的应用程序开发。

第四章—应用程序说明，介绍了提供给用户的应用程序的使用方法，通过这个应用程序，用户可以不需编程而直接进行一些基本的通讯操作，达到快速应用的目的。另外，在该章中，也提供了驱动接口使用例程，为用户开发适合个人要求的应用程序的提供参考。

# 第一章　概述

## 1.1 功能

- PCI 数据接口总线
- 1553
  - 1 个 BC（总线控制器）
  - 0~31 个 RT（远程终端）
  - 1 个 MT（总线监视器）
  - 自动 BC 重试
  - BC 支持帧重复发送
  - 可设置帧重复发送的次数：有限次发送和无限次重复发送
  - 可设置 BC 帧间隔时间和消息间隔时间
  - 支持时标模式
  - RT 方式下可设置非法命令表
  - MT 方式下支持过滤功能
  - 双通道数据发送和接收
  - 大容量的数据存储：16M × 16bit
  - 8 路 TTL 数字量输入和输出
- 串口
  - 防浪涌保护及光电隔离
  - 4 路独立的异步 RS232/422/485 全双工或半双工通信
  - 波特率 400～4Mbps 可设置
  - 数据位长度、校验位、停止位可设置
  - 两种数据接收方式：透明接收和协议接收
  - 协议接收时，可设置接收帧头及帧长度
  - 提供 FIFO 空、满标志
  - 接收 FIFO 共 32MByte
  - 每路发送通道有 2KByte 的发送 FIFO

- ARINC429
  - 4 路发送通道，每路带 256x32-Bit FIFO
  - 4 路接收通道, 每路带 512x32-Bit FIFO
  - 具有定时发送功能
  - 标号过滤功能
  - 添加时间标签功能
  - FIFO 触发深度可设功能
  - 波特率 100K、48K、12.5K 可设置等功能
  - 中断支持

## 1.2 基本工作环境

操作系统：Windows 98/2000/NT/XP

工作温度：-40 ~ +85℃

相对湿度：0 ~ 95％

## 1.3 安装

硬件安装：将板卡插入 PCI 主机箱的设备插槽（插槽有板卡导槽并且有防反插功能），直至将板卡完全插入插槽。

软件安装：见 第三章 驱动接口说明。

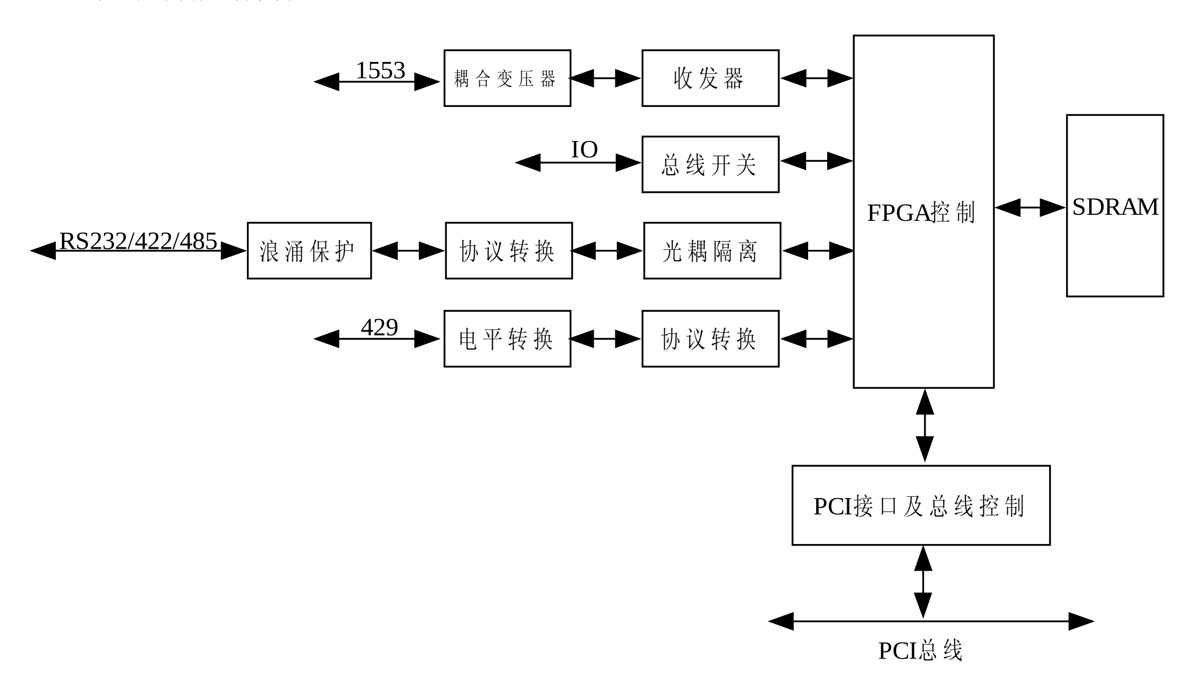
# 第二章　硬件使用说明

## 2.1 功能说明

- PCI 数据接口总线
- 1553
  - 1 个 BC（总线控制器）
  - 0~31 个 RT（远程终端）
  - 1 个 MT（总线监视器）
  - 自动 BC 重试
  - BC 支持帧重复发送
  - 可设置帧重复发送的次数：有限次发送和无限次重复发送
  - 可设置 BC 帧间隔时间和消息间隔时间
  - 支持时标模式
  - RT 方式下可设置非法命令表
  - MT 方式下支持过滤功能
  - 双通道数据发送和接收
  - 大容量的数据存储：16M × 16bit
  - 8 路 TTL 数字量输入和输出
- 串口
  - 防浪涌保护及光电隔离
  - 4 路独立的异步 RS232/422/485 全双工或半双工通信
  - 波特率 400～4Mbps 可设置
  - 数据位长度、校验位、停止位可设置
  - 两种数据接收方式：透明接收和协议接收
  - 协议接收时，可设置接收帧头及帧长度
  - 提供 FIFO 空、满标志
  - 接收 FIFO 共 32MByte
  - 每路发送通道有 2KByte 的发送 FIFO

- ARINC429
  - 4 路发送通道，每路带 256x32-Bit FIFO
  - 4 路接收通道，每路带 512x32-Bit FIFO
  - 具有定时发送功能
  - 标号过滤功能
  - 添加时间标签功能
  - FIFO 触发深度可设功能
  - 波特率 100K、48K、12.5K 可设置等功能
  - 中断支持

板卡功能结构图




## 2.2 配置与安装

### 2.2.1. 配置方法

本板卡在使用之前不需要进行硬件配置，所有的初始化都由软件操作。

### 2.2.2. 安装步骤

本板卡为采用标准 PCI 短卡尺寸结构，将卡插入 PCI 插槽，用螺丝钉将板卡的挡板与机箱固定在一起即可。

## 2.3 通讯接口定义

板卡采用 SISC68（公头）连接器，点定义如下：

| 点号 | 名称 | 说明 | 点号 | 名称 | 说明 |
|---|---|---|---|---|---|
| 1 | OUT1- | 第1路<br>422、485全双工状态输出-<br>（422、485半双工状态输入-）<br>或232输出 | 35 | IN2+ | 第2路<br>422、485全双工状态输入+<br>或232输入 |
| 2 | IN1- | 第1路<br>422、485全双工状态输入- | 36 | OUT1+ | 第1路<br>422、485全双工状态输出+<br>（422、485半双工状态输入+） |
| 3 | IN1+ | 第1路<br>422、485全双工状态输入+<br>或232输入 | 37 | IN2- | 第2路<br>422、485全双工状态输入- |
| 4 | OUT2+ | 第2路<br>422、485全双工状态输出+<br>（422、485半双工状态输入+） | 38 | GD1 | 第1路参考地 |
| 5 | GD1 | 第1路参考地 | 39 | GD2 | 第2路参考地 |
| 6 | IN4+ | 第4路<br>422、485全双工状态输入+<br>或232输入 | 40 | OUT2- | 第2路<br>422、485全双工状态输出-<br>（422、485半双工状态输入-）<br>或232输出 |
| 7 | IN4- | 第4路<br>RS422、485全双工状态输入- | 41 | GD2 | 第2路参考地 |
| 8 | OUT3+ | 第3路<br>422、485全双工状态输出+<br>（422、485半双工状态输入+） | 42 | GD4 | 第4路参考地 |
| 9 | OUT3- | 第3路<br>422、485全双工状态输出-<br>（422、485半双工状态输入-）<br>或232输出 | 43 | GD3 | 第3路参考地 |
| 10 | IN3- | 第3路<br>422、485全双工状态输入- | 44 | IN3+ | 第3路<br>422、485全双工状态输入+<br>或232输入 |
| 11 | OUT4- | 第4路<br>422、485全双工状态输出-<br>（422、485半双工状态输入-）<br>或232输出 | 45 | GD3 | 第3路参考地 |
| 12 | OUT4+ | 第4路 | 46 | GD4 | 第4路参考地 |

| | | 422、485全双工状态输出+<br>（422、485半双工状态输入+） | | | |
|---|---|---|---|---|---|
| 13 | TTLDO7 | TTL数字量输出（DO7） | 47 | TTLDO6 | TTL数字量输出（DO6） |
| 14 | TTLDO5 | TTL数字量输出（DO5） | 48 | TTLDO4 | TTL数字量输出（DO4） |
| 15 | TTLDO3 | TTL数字量输出（DO3） | 49 | TTLDO2 | TTL数字量输出（DO2） |
| 16 | TTLDO1 | TTL数字量输出（DO1） | 50 | TTLDO0 | TTL数字量输出（DO0） |
| 17 | TTLDI7 | TTL数字量输入（DI7） | 51 | TTLDI6 | TTL数字量输入（DI6） |
| 18 | TTLDI5 | TTL数字量输入（DI5） | 52 | TTLDI4 | TTL数字量输入（DI4） |
| 19 | TTLDI3 | TTL数字量输入（DI3） | 53 | TTLDI2 | TTL数字量输入（DI2） |
| 20 | TTLDI1 | TTL数字量输入（DI1） | 54 | TTLDI0 | TTL数字量输入（DI0） |
| 21 | | | 55 | | |
| 22 | OUT4+ | 第四路429输出+ | 56 | OUT4- | 第四路429输出- |
| 23 | OUT2- | 第二路429输出- | 57 | OUT2+ | 第二路429输出+ |
| 24 | IN4- | 第四路429输入- | 58 | IN4+ | 第四路429输入+ |
| 25 | IN3- | 第三路429输入- | 59 | IN3+ | 第三路429输入+ |
| 26 | IN2- | 第二路429输入- | 60 | IN2+ | 第二路429输入+ |
| 27 | IN1- | 第一路429输入- | 61 | IN1+ | 第一路429输入+ |
| 28 | OUT1+ | 第一路429输出+ | 62 | OUT1- | 第一路429输出- |
| 29 | OUT3- | 第三路429输出- | 63 | OUT3+ | 第三路429输出+ |
| 30 | | | 64 | | |
| 31 | CHB_C- | B通道1553总线直接偶合- | 65 | CHB_C+ | B通道1553总线直接偶合+ |
| 32 | CHB_D- | B通道1553总线间接偶合- | 66 | CHB_D+ | B通道1553总线间接偶合+ |
| 33 | CHA_C- | A通道1553总线直接偶合- | 67 | CHA_C+ | A通道1553总线直接偶合+ |
| 34 | CHA_D- | A通道1553总线间接偶合- | 68 | CHA_D+ | A通道1553总线间接偶合+ |

| 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | ...... | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ...... | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |

板卡上SCSI 68接头点号示意图

# 第三章　驱动接口说明

## 3.1. 主要功能

- PCI 数据接口总线
- 1553
  - 1 个 BC（总线控制器）
  - 0~31 个 RT（远程终端）
  - 1 个 MT（总线监视器）
  - 自动 BC 重试
  - BC 支持帧重复发送
  - 可设置帧重复发送的次数：有限次发送和无限次重复发送
  - 可设置 BC 帧间隔时间和消息间隔时间
  - 支持时标模式
  - RT 方式下可设置非法命令表
  - MT 方式下支持过滤功能
  - 双通道数据发送和接收
  - 大容量的数据存储：16M × 16bit
  - 8 路 TTL 数字量输入和输出
- 串口
  - 防浪涌保护及光电隔离
  - 4 路独立的异步 RS232/422/485 全双工或半双工通信
  - 波特率 400～4Mbps 可设置
  - 数据位长度、校验位、停止位可设置
  - 两种数据接收方式：透明接收和协议接收
  - 协议接收时，可设置接收帧头及帧长度
  - 提供 FIFO 空、满标志
  - 接收 FIFO 共 32MByte
  - 每路发送通道有 2KByte 的发送 FIFO

- ARINC429
  - 4 路发送通道，每路带 256x32-Bit FIFO
  - 4 路接收通道, 每路带 512x32-Bit FIFO
  - 具有定时发送功能
  - 标号过滤功能
  - 添加时间标签功能
  - FIFO 触发深度可设功能
  - 波特率 100K、48K、12.5K 可设置等功能
  - 中断支持

## 3.2. 驱动运行环境

Windows 98/2000/xp 操作系统

## 3.3. 驱动安装说明

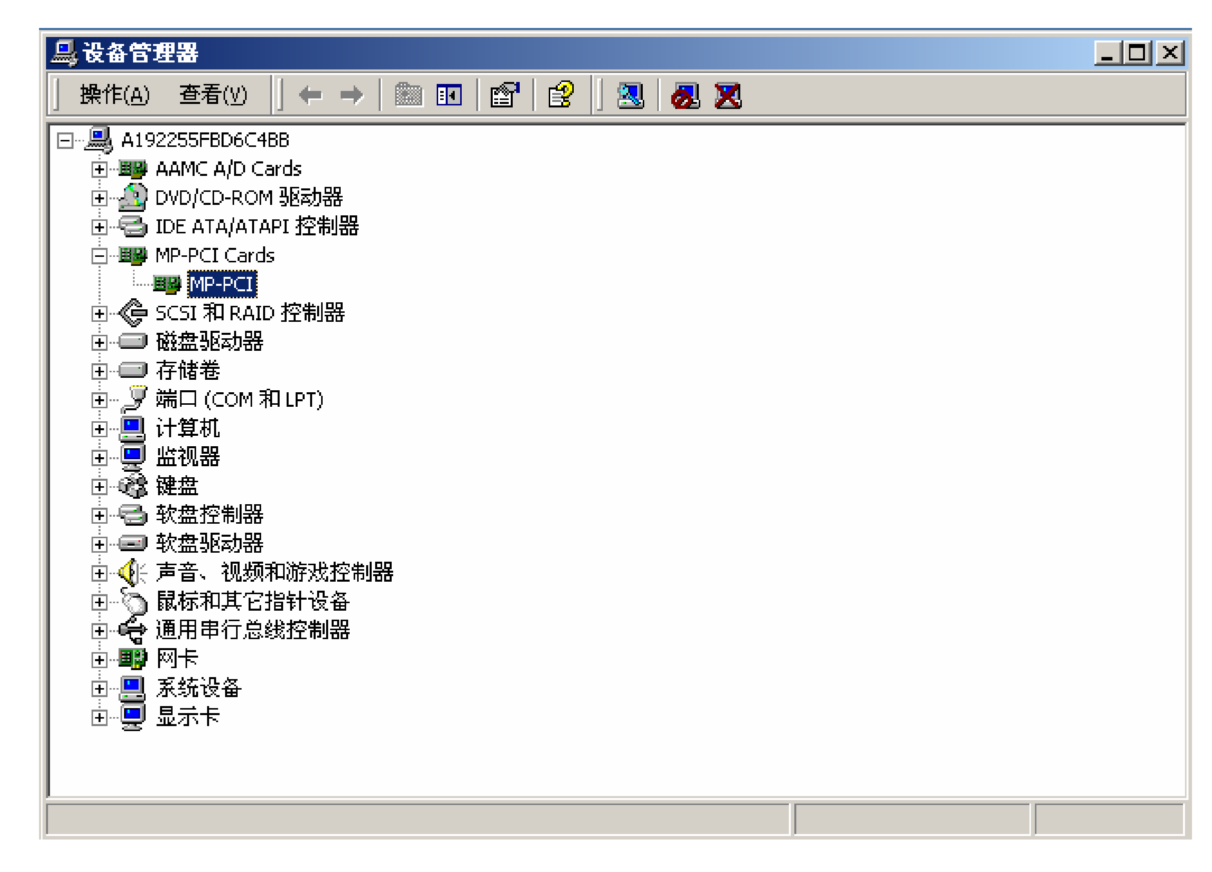
1、 将板卡插入PC机PCI插槽;

2、 系统提示找到新硬件，需要安装它的软件;

3、 点击硬件安装的下一步，若PC机是Windows 2000/98的操作系统，请指向光盘中“驱动安装文件\driver4win2k”目录；若PC机是Windows XP操作系统，请指向光盘中“驱动安装文件\driver4winxp”目录，选择MP_PCI.inf文件。

4、 点击下一步，直到驱动程序安装成功。

5、 安装完毕，在设备管理器会看到如下信息（下图蓝色区域）




## 3.4. 驱动程序使用说明

### 3.4.1 驱动程序引用文件

6、 库文件：MP_PCI.dll 和 MP_PCI.lib

7、 函数库头文件：MP_PCI_lib.h

defines.h

### 3.4.2 驱动程序引用的结构

#### 3.4.2.1 1553 部分

**1、 BC 消息发送结构**

```
typedef struct
{
   BOOL RetryEnable;
   BYTE ChannelSelect;
   DWORD InterMSGGapTime;
```

```
    BYTE MSGFormat;
    WORD MSGBlock[37];
}SMSG_STRUCT;
```

结构参数说明：

**RetryEnable**：消息重试允许位　TRUE：允许消息重试

**ChannelSelect**：消息发送的通道选择 0:Channel B　1:Channel A

**InterMSGGapTime**：设置消息间的间隔，单位 1μs

**MSGFormat**：设置消息的格式，消息的格式设置见表 1

**MSGBlock**：存放待发送的消息，消息的存放格式见表 2

**表 1**

| BIT7~3 | BIT2 | BIT1 | BIT 0 | 消 息 格 式 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | BC-to-RT（如果 T/$\overline{R}$=0 ）或者 RT-to-BC (如果 T/$\overline{R}$=1) |
| 0 | 0 | 0 | 1 | RT-to-RT |
| 0 | 0 | 1 | 0 | Broadcast |
| 0 | 0 | 1 | 1 | RT-to-RTs （Broadcast） |
| 0 | 1 | 0 | 0 | Mode Code |
| 0 | 1 | 0 | 1 | 保留 |
| 0 | 1 | 1 | 0 | Broadcast Mode Code |
| 0 | 1 | 1 | 1 | 保留 |

**表 2**



| BC-to-RT传输 |
| --- |
| BC控制字 |
| 接收命令字 |
| 数据字#1 |
| 数据字#2 |
| ... |
| 最后一个数据字 |
| 返回的最后一个数据字 |
| 接收到的状态字 |

| RT-to-BC传输 |
| --- |
| BC控制字 |
| 发送命令字 |
| 返回的发送命令字 |
| 接收到的状态字 |
| 接收到的数据字#1 |
| 接收到的数据字#2 |
| ... |
| 接收到最后一个数据字 |

| RT-to-RT传输 |
| --- |
| BC控制字 |
| 接收命令字 |
| 发送命令字 |
| 返回的发送命令字 |
| 发送RT状态字 |
| 数据字#1 |
| 数据字#2 |
| ... |
| 最后一个数据 |
| 接收RT状态字 |

| 不带数据的模式代码 |
| --- |
| BC控制字 |
| 模式代码命令 |
| 返回的模式代码命令 |
| 接收到的状态字 |

| 发送带数据的模式代码 |
| --- |
| BC控制字 |
| 发送模式代码命令 |
| 返回的模式代码命令 |
| 接收到的状态字 |
| 数据字 |

| 接收带数据的模式代码 |
| --- |
| BC控制字 |
| 接收模式代码命令 |
| 数据字 |
| 返回的数据字 |
| 接收到的状态字 |

| 广播 |
| --- |
| BC控制字 |
| 广播命令字 |
| 数据#1 |
| 数据#2 |
| ... |
| 最后一个数据字 |
| 返回的最后一个数据 |

| RT-to-RTs（广播）传输 |
| --- |
| BC控制字 |
| 接收广播命令字 |
| 发送命令 |
| 返回的发送命令 |
| 发送RT状态字 |
| 数据字#1 |
| 数据字#2 |
| ... |
| 最后一个数据字 |

| 不带数据的广播模式代码 |
| --- |
| BC控制字 |
| 广播模式代码命令 |
| 返回的广播模式代码命令 |

| 带数据的广播模式代码 |
| --- |
| BC控制字 |
| 广播模式代码命令 |
| 数据字 |
| 返回的数据字 |

## 2、 消息接收结构

```
typedef struct
{
    WORD BSW;-----------------  消息状态描述字
    DWORD TimeTag;----------  时标，单位 1μs
    WORD MSGBlock[37];----  存放消息，消息的存放结构同表 2
}RMSG_STRUCT;
```

**结构参数说明：**

**BSW：** Block Status Word 消息块状态描述字，格式见表 3

**TimeTag：** 时标，在时标模式不启用时，该项无意义，为 0

**MSGBlock**：存放接收到的消息，BC 消息的存放结构同表 2；RT 消息第 0 个字节存放的为接收到的命令字，接下来的字节为接收到的数据；MT 消息的存放结构见表 4

**表 3**

**BC 消息块状态描述字：**

| BIT | Description | |
|---|---|---|
| 15 | EOM | BC 消息结束标志，1 有效 |
| 14 | SOM | BC 消息开始标志，1 有效 |
| 13 | CHANNEL B/A | 消息发送通道指示 0：消息从 A 通道发出 1：消息从 B 通道发出 |
| 12 | ERROR FLAG | 消息出错指示，当出现 8、9、10 位中的任何一种情况时，1 有效 |
| 11 | STATUS SET | 状态字低 11 位存在非 0 的位，1 有效 |
| 10 | FORMAT ERROR | 帧格式错误，1 有效 |
| 9 | NO RESPONSE TIMEOUT | 应答超时，1 有效 |
| 8 | CMD_ERR | 发送的命令有误，1 有效 |
| 7 | RT-RT FORMAT | RT-RT 消息指示，1 有效 |
| 6 | RETRY COUNT 1 | 重试次数 0：0 次 1：1 次 3：2 次 |
| 5 | RETRY COUNT 0 | |
| 4 | GOOD DATA BLOCK TRANSFER | 消息传送正常指示，1 有效 |
| 3 | AD_ERR | RT 状态字地址有误，1 有效 |
| 2 | LEN_ERR | 字长有误，1 有效 |
| 1 | INCORRECT SYNC TYPE | 同步字头错误，1 有效 |

| 0 | INVALID WORD | 同步字、曼彻斯特编码、校验或位长度有误，1 有效 |
|---|---|---|

**RT 消息块状态描述字:**

| BIT | Description | |
|---|---|---|
| 15 | 保留 | |
| 14 | 保留 | |
| 13 | CHANNEL B/A | 消息发送通道指示 0：消息从 A 通道发出 1：消息从 B 通道发出 |
| 12 | ERROR FLAG | 消息出错指示，当出现 8、9、10 位中的任何一种情况时，1 有效 |
| 11 | RT-RT FORMAT | RT-RT 消息指示，1 有效 |
| 10 | FORMAT ERROR | 帧格式错误，1 有效 |
| 9 | NO RESPONSE TIMEOUT | 应答超时，1 有效 |
| 8 | 保留 | |
| 7 | ROB | 循环缓冲模式下缓冲区溢出标志 |
| 6 | CMDILL | 非法指令指示 |
| 5 | WL_ERR | 数据长度错误标志 |
| 4 | SYN_ERR | 同步字头错误标志 |
| 3 | WD_ERR | 数据错误标志 |
| 2 | AD_ERR | 状态字地址错误标志 |
| 1 | CMD2_ERR | RT-RT 传输中第二个命令字错误标志 |
| 0 | CMD_ERR | 命令字错误标志 |

**MT 消息块状态描述字:**

| BIT | Description | |
|---|---|---|
| 15 | EOM | 消息结束标志，1 有效 |

| 14 | SOM | 消息开始标志，1 有效 |
|---|---|---|
| 13 | CHANNEL B/A | 消息发送通道指示 0：消息从 A 通道发出 1：消息从 B 通道发出 |
| 12 | ERROR FLAG | 消息出错指示，当出现 8、9、10 位中的任何一种情况时，高有效 |
| 11 | RT-RT FORMAT | RT-RT 消息指示，1 有效 |
| 10 | FORMAT ERROR | 帧格式错误，1 有效 |
| 9 | NO RESPONSE TIMEOUT | 应答超时，1 有效 |
| 8 | 保留 | |
| 7 | GOOD | 消息传送正常指示，1 有效 |
| 6 | ROB | 循环缓冲模式下缓冲区溢出标志 |
| 5 | LEN_ERR | 数据长度错误标志 |
| 4 | SYN_ERR | 同步字头错误标志 |
| 3 | WD_ERR | 数据错误标志 |
| 2 | AD_ERR | 状态字地址错误标志 |
| 1 | CMD2_ERR | RT-RT 传输中第二个命令字错误标志 |
| 0 | CMD_ERR | 命令字错误标志 |

表 4

| BC-to-RT传输 |
|---|
| BC控制字 |
| 接收命令字 |
| 数据字#1 |
| 数据字#2 |
| ... |
| 最后一个数据字 |
| 返回的最后一个数据字 |
| 接收到的状态字 |

| RT-to-BC传输 |
|---|
| BC控制字 |
| 发送命令字 |
| 返回的发送命令字 |
| 接收到的状态字 |
| 接收到的数据字#1 |
| 接收到的数据字#2 |
| ... |
| 接收到最后一个数据字 |

| RT-to-RT传输 |
|---|
| BC控制字 |
| 接收命令字 |
| 发送命令字 |
| 返回的发送命令字 |
| 发送RT状态字 |
| 数据字#1 |
| 数据字#2 |
| ... |
| 最后一个数据 |
| 接收RT状态字 |

| 不带数据的模式代码 |
|---|
| BC控制字 |
| 模式代码命令 |
| 返回的模式代码命令 |
| 接收到的状态字 |

| 发送带数据的模式代码 |
|---|
| BC控制字 |
| 发送模式代码命令 |
| 返回的模式代码命令 |
| 接收到的状态字 |
| 数据字 |

| 接收带数据的模式代码 |
|---|
| BC控制字 |
| 接收模式代码命令 |
| 数据字 |
| 返回的数据字 |
| 接收到的状态字 |

| 广播 |
|---|
| BC控制字 |
| 广播命令字 |
| 数据#1 |
| 数据#2 |
| ... |
| 最后一个数据字 |
| 返回的最后一个数据 |

| RT-to-RTs（广播）传输 |
|---|
| BC控制字 |
| 接收广播命令字 |
| 发送命令 |
| 返回的发送命令 |
| 发送RT状态字 |
| 数据字#1 |
| 数据字#2 |
| ... |
| 最后一个数据字 |

| 不带数据的广播模式代码 |
|---|
| BC控制字 |
| 广播模式代码命令 |
| 返回的广播模式代码命令 |

| 带数据的广播模式代码 |
|---|
| BC控制字 |
| 广播模式代码命令 |
| 数据字 |
| 返回的数据字 |

## 3、 帧发送结构

```
typedef struct
{
    WORD MSGNum;-------------------------待发送消息的数目，最大为 4096
    SMSG_STRUCT SelfMSG[4096];------存放待发送的消息
}SFRAME_STRUCT;
```

## 4、 帧接收结构

```
typedef struct
{
    WORD MSGNum;-------------------存放接收到的消息的数目
    RMSG_STRUCT RMSG[4096]---存放接收到的消息，数组的有效长度为 MSGNum
}RFRAME_STRUCT;
```

## 5、 消息重试结构

```
typedef struct
{
    BOOL Retry_IF_MSGErr;
    BOOL Retry_IF_StatusSet;
}RETRY_CASE_STRUCT;
```

**结构参数说明：**

除了应答超时、格式错误（状态字地址错误，数据格式错误，同步字错误等）消息会重试外，以下 2 种情况也可使消息重试：

**Retry_IF_1553A_MSGErr**：RT 状态字中的 Message Error 位为 1

**Retry_IF_StatusSet：RT 的状态字被置位**

## 6、 消息重试通道选择

```
typedef struct
{
    BOOL Alter_Chan_On_Busy1;
    BOOL Alter_Chan_On_Busy2;
}RETRY_CHANNEL_SEL_STRUCT;
```

结构参数说明：

**Alter_Chan_On_Busy1**：第一次重试改变通道

TRUE：如果消息第一次传输是在通道 A，那么第一次重试是改为 B 通道传输；

FALSE：不改变通道

**Alter_Chan_On_Busy2**：第二次重试改变通道

TRUE：如果消息第一次传输是在通道 A，那么第二次重试是改为 B 通道传输；

FALSE：不改变通道

## 7、 BC 消息和帧处理当 RT STATUS_SET 置位

```
typedef struct
{
    BOOL Stop_On_MSG;
    BOOL Stop_On_Frame;
}STATUS_SET_STRUCT;
```

结构参数说明：

**Stop_On_MSG**：TRUE----如果 RT 状态字中 STATUS_SET 置位，在处理完本条消息后将停止消息处理

**Stop_On_Frame**：TRUE----如果 RT 状态字中 STATUS_SET 置位，在处理完本帧后将停止帧处理

## 8、 BC 消息处理当消息出错

```
typedef struct
{
    BOOL MSG_STOP_ON_ERR;
    BOOL FRAME_STOP_ON_ERR;
}STOP_ON_ERR_STRUCT;
```

结构参数说明：

**MSG_STOP_ON_ERR**：TRUE----消息出错(包括字错误、帧格式错误、超时错误)时停止消息处理，但如果重试使能，那么先重试，重试还有错误再停止

**FRAME_STOP_ON_ERR**：TRUE----在自动重发模式下，出错时消息帧停止

## 9、 RT STATUS 置位

```
typedef struct
{
    BOOL TerminalFlag;
    BOOL SubSystemFlag;
    BOOL ServiceReq;
    BOOL Busy;
    BOOL DBusCtl;
}RT_STATUS_WORD_STRUCT;]
```

结构参数说明：

**TerminalFlag**： 如果该位为真，RT的状态字中的“Terminal Flag”位将会置1

**SubSystemFlag**：如果该位为真，RT 的状态字中的“Subsystem Flag”位将会置 1

**ServiceReq**：如果该位为真，RT 的状态字中的“Service Request”位将会置 1

**Busy**：如果该位为真，RT 的状态字中的“Busy”位将会置 1

**DbusCtl**：如果该位为真，RT 的状态字中的“Dynamic Bus Control Acceptance”位将会置 1

## 10、 RT 非法命令表结构

```
typedef struct
{
    DWORD CmdTable[32][2][32];
}RT_Illegal_CMD_TABLE_STRUCT;
```

结构参数说明：

**CmdTable[I][J][K]**：一个三维的 RT 非法命令表，一维坐标 I 代表 RT 的地址，二位坐标 J 代表发送或接收位（J=0，代表接收 J=1，代表发送），三维坐标 K 代表 RT 的子地址。数组的值的定义如下：

| BITS | DESCRIPTION |
|---|---|
| D0 | 1：数据量为 32 的命令字非法 0：合法 |
| D1 | 1：数据量为 1 的命令字非法 0：合法 |
| D2 | 1：数据量为 2 的命令字非法 0：合法 |
| D3 | 1：数据量为 3 的命令字非法 0：合法 |
| D4 | 1：数据量为 4 的命令字非法 0：合法 |
| D5 | 1：数据量为 5 的命令字非法 0：合法 |
| D6 | 1：数据量为 6 的命令字非法 0：合法 |
| D7 | 1：数据量为 7 的命令字非法 0：合法 |
| D8 | 1：数据量为 8 的命令字非法 0：合法 |
| D9 | 1：数据量为 9 的命令字非法 0：合法 |
| D10 | 1：数据量为 10 的命令字非法 0：合法 |

| D11 | 1：数据量为 11 的命令字非法 0：合法 |
|---|---|
| D12 | 1：数据量为 12 的命令字非法 0：合法 |
| D13 | 1：数据量为 13 的命令字非法 0：合法 |
| D14 | 1：数据量为 14 的命令字非法 0：合法 |
| D15 | 1：数据量为 15 的命令字非法 0：合法 |
| D16 | 1：数据量为 16 的命令字非法 0：合法 |
| D17 | 1：数据量为 17 的命令字非法 0：合法 |
| D18 | 1：数据量为 18 的命令字非法 0：合法 |
| D19 | 1：数据量为 19 的命令字非法 0：合法 |
| D20 | 1：数据量为 20 的命令字非法 0：合法 |
| D21 | 1：数据量为 21 的命令字非法 0：合法 |
| D22 | 1：数据量为 22 的命令字非法 0：合法 |
| D23 | 1：数据量为 23 的命令字非法 0：合法 |
| D24 | 1：数据量为 24 的命令字非法 0：合法 |
| D25 | 1：数据量为 25 的命令字非法 0：合法 |
| D26 | 1：数据量为 26 的命令字非法 0：合法 |
| D27 | 1：数据量为 27 的命令字非法 0：合法 |
| D28 | 1：数据量为 28 的命令字非法 0：合法 |
| D29 | 1：数据量为 29 的命令字非法 0：合法 |
| D30 | 1：数据量为 30 的命令字非法 0：合法 |
| D31 | 1：数据量为 31 的命令字非法 0：合法 |

Eg：CmdTable[3][1][20] = 0x00000001 表示 RT 地址为 3，RT 的子地址为 20，发送 32 个数据的命令为非法的。

## 11、 RT 发送模式设置结构

typedef struct

```
{
   BYTE TxMode[32][32];
}RT_TX_MODE_STRUCT;
```

**结构参数说明：**

**TxMode[I][J]**：一个二维的模式设置数组，数组的行坐标代表 RT 的地址，数组的列坐标代表 RT 的子地址，数组的值为模式设置位，取值 0 为单缓冲模式，此种模式下 RT 发送数据都从发送数据区起始点读取数据发送；取值 1 为循环缓冲模式，此种模式下 RT 顺序从发送区读取数据发送，直到发送数据区指针到达设定的边界时，发送指针自动回到起始位置继续读数用于发送。

Eg：TxMode[1][6] = 1 表示 RT 地址为 1，子地址为 6 的远程终端的发送采用循环缓冲的模式

## 12、 MT 命令字过滤表结构

```
typedef struct
{
    DWORD Filter[32][2];
}MT_CMD_FILTER_TABLE_STRUCT;
```

结构参数说明：

Filter[I][J]：一个二维的命令字过滤表，数组的行坐标 I 代表待监测的远程终端地址，数组的列坐标代表发送或接收位（J=0：接收　J=1：发送），数组的值定义如下：

| BITS | DESCRIPTION |
|---|---|
| D0 | 1：方式代码被监测　0：不被监测，消息丢掉 |
| D1 | 1：子地址为 1 的远程终端被监测　0：不被监测，消息丢掉 |
| D2 | 1：子地址为 2 的远程终端被监测　0：不被监测，消息丢掉 |

| | |
|---|---|
| D3 | 1：子地址为 3 的远程终端被监测　　0：不被监测，消息丢掉 |
| D4 | 1：子地址为 4 的远程终端被监测　　0：不被监测，消息丢掉 |
| D5 | 1：子地址为 5 的远程终端被监测　　0：不被监测，消息丢掉 |
| D6 | 1：子地址为 6 的远程终端被监测　　0：不被监测，消息丢掉 |
| D7 | 1：子地址为 7 的远程终端被监测　　0：不被监测，消息丢掉 |
| D8 | 1：子地址为 8 的远程终端被监测　　0：不被监测，消息丢掉 |
| D9 | 1：子地址为 9 的远程终端被监测　　0：不被监测，消息丢掉 |
| D10 | 1：子地址为 10 的远程终端被监测　　0：不被监测，消息丢掉 |
| D11 | 1：子地址为 11 的远程终端被监测　　0：不被监测，消息丢掉 |
| D12 | 1：子地址为 12 的远程终端被监测　　0：不被监测，消息丢掉 |
| D13 | 1：子地址为 13 的远程终端被监测　　0：不被监测，消息丢掉 |
| D14 | 1：子地址为 14 的远程终端被监测　　0：不被监测，消息丢掉 |
| D15 | 1：子地址为 15 的远程终端被监测　　0：不被监测，消息丢掉 |
| D16 | 1：子地址为 16 的远程终端被监测　　0：不被监测，消息丢掉 |
| D17 | 1：子地址为 17 的远程终端被监测　　0：不被监测，消息丢掉 |
| D18 | 1：子地址为 18 的远程终端被监测　　0：不被监测，消息丢掉 |
| D19 | 1：子地址为 19 的远程终端被监测　　0：不被监测，消息丢掉 |
| D20 | 1：子地址为 20 的远程终端被监测　　0：不被监测，消息丢掉 |
| D21 | 1：子地址为 21 的远程终端被监测　　0：不被监测，消息丢掉 |
| D22 | 1：子地址为 22 的远程终端被监测　　0：不被监测，消息丢掉 |
| D23 | 1：子地址为 23 的远程终端被监测　　0：不被监测，消息丢掉 |
| D24 | 1：子地址为 24 的远程终端被监测　　0：不被监测，消息丢掉 |
| D25 | 1：子地址为 25 的远程终端被监测　　0：不被监测，消息丢掉 |
| D26 | 1：子地址为 26 的远程终端被监测　　0：不被监测，消息丢掉 |
| D27 | 1：子地址为 27 的远程终端被监测　　0：不被监测，消息丢掉 |
| D28 | 1：子地址为 28 的远程终端被监测　　0：不被监测，消息丢掉 |
| D29 | 1：子地址为 29 的远程终端被监测　　0：不被监测，消息丢掉 |

| D30 | 1：子地址为 30 的远程终端被监测　0：不被监测，消息丢掉 |
| --- | --- |
| D31 | 1：方式代码被监测　0：不被监测，消息丢掉 |

### 3.4.2.2 ARINC429 部分

**1、 FIFO 触发深度设置结构**

```
typedef struct TriggerDepth
{
    WORD Chan0Depth_I;--------第 1 路接收 FIFO 触发深度寄存器
    WORD Chan1Depth_I;--------第 2 路接收 FIFO 触发深度寄存器
    WORD Chan2Depth_I;--------第 3 路接收 FIFO 触发深度寄存器
    WORD Chan3Depth_I;--------第 4S 路接收 FIFO 触发深度寄存器
    BYTE Chan0Depth_O;-------第 1 路发送 FIFO 触发深度寄存器
    BYTE Chan1Depth_O;-------第 2 路发送 FIFO 触发深度寄存器
    BYTE Chan2Depth_O;-------第 3 路发送 FIFO 触发深度寄存器
    BYTE Chan3Depth_O;-------第 4 路发送 FIFO 触发深度寄存器
} TriggerDepth_STRUCT;
```

注：4 路接收 FIFO 的触发深度最大值为 510，4 路发送 FIFO 的触发深度最大值为 254，收 FIFO 里的数据量大于设置的触发深度值，则产生一个触发标志；若发送 FIFO 里的数据量小于设置的触发深度值，则产生一个触发标志。

**2、 标号过滤表设置结构**

```
typedef struct LabelTable
{
    BYTE LabFilterChan0 [4][256];------第 1 路标号过滤表设置
    BYTE LabFilterChan1 [4][256];------第 2 路标号过滤表设置
    BYTE LabFilterChan2 [4][256];------第 3 路标号过滤表设置
    BYTE LabFilterChan3 [4][256];------第 4 路标号过路表设置
```

} LabelTable_STRUCT;

注：LabelFilterChanX：数组的行坐标值代表 S/D 号，列坐标值代表标号；若数组对应元素的值不为零，则 S/D 号对应的标号参与过滤。例：LabelFilterChanX[1][2]=1：表示 S/D 为 1 且 Label 为 2 的 ARINC429 数据参与过滤，即被接收

### 3.4.3 驱动程序函数接口说明

#### 3.4.3.1 打开和关闭板卡

**1、 MPCard_Open**

函数原型：BOOL __stdcall MPCard_Open (HANDLE *phMPCard,
BYTE CardId);

函数功能：找板卡，并分配板卡资源

参数说明：phMPCard：板卡句柄的指针

CardId：板卡编号，取值为 0~255（若 PC 机中同时插 256 块 AAMC-PCI-MP 板卡，板卡的编号按板卡所在的插槽离 CPU 的距离由近到远以次编号 0，1,..,255；若 PC 机中同时只插一块板，板卡编号为 0）

返回值： 若板卡打开成功，返回值为真；否则为假

**2、 MPCard_Close**

函数原型：BOOL __stdcall **MPCard_Close** (HANDLE hMPCard);

函数功能：关闭板卡，释放板卡资源

参数说明：hMPCard：板卡的句柄

返回值： 若板卡关闭成功，返回值为真；否则为假

#### 3.4.3.2 1553 部分

**1、 MP1553_Reset**

函数原型：BOOL __stdcall **MP1553_Reset** (HANDLE hMPCard);

函数功能：板卡 1553 部分复位函数

**参数说明**：hMPCard：板卡的句柄

**返回值**：　若板卡复位成功，返回值为真；否则为假

**说明**：板卡复位执行的操作为：

BC 停止

31 个 RT 不使能

MT 不使能

时间标签模式不使能

BC 消息到达标志和帧到达标志被清除

BC 两次重试不改变通道

BC 重试不使能

BC 消息在 Message Error 和 RT STATUS_SET 置位时不重试

RT STATUS_SET 置位后 BC 消息和帧发送不停止

消息出错时 BC 消息和帧发送不停止

BC 帧重复发送模式不使能

BC 帧间间隔清零

应答超时设置清零

RT 非法命令表不使能

RT 接收非法命令数据

其余的设置保持不变

**2、 MP1553_AddTimeTag**

**函数原型**：BOOL __stdcall **MP1553_AddTimeTag** (HANDLE hMPCard,

BOOL Enable);

**函数功能**：启动或停止时间标签模式

**参数说明**：hMPCard：板卡的句柄

Enable：时标使能位　TRUE：时标使能

**返回值**：　若时标模式设置成功，返回值为真；否则为假

**3、 MP1553_SetResponseTimeout**

函数原型：BOOL __stdcall MP1553_SetResponseTimeout (HANDLE hMPCard,WORD TimeOut);

函数功能：设置应答超时

参数说明：hMPCard：板卡的句柄

TimeOut：超时时间设置寄存器 单位 0.5μS

返回值： 若设置成功，返回值为真；否则为假

**4、 BC_Init**

函数原型：void __stdcall **BC_Init** (HANDLE hMPCard);

函数功能：初始化总线控制器

参数说明：hMPCard：板卡的句柄

返回值： 无

**5、 BC_SetFrameGap**

函数原型：BOOL __stdcall **BC_SetFrameGap** (HANDLE hMPCard, DWORD Gap);

函数功能：设置帧间隔时间

参数说明：hMPCard：板卡的句柄

Gap：帧间间隔设置寄存器，单位 1μS

返回值： 若设置成功，返回值为真；否则为假

**6、 BC_FrameAutoRepeat**

函数原型：BOOL __stdcall **BC_FrameAutoRepeat** (HANDLE hMPCard, BOOL Enable);

函数功能：帧自动重发模式设置

参数说明：hMPCard：板卡的句柄

Enable：使能位 TRUE：帧自动重发使能

返回值： 若设置成功，返回值为真；否则为假

**7、 BC_FrameAutoRepeat_Count**

函数原型：BOOL __stdcall **BC_FrameAutoRepeat_Count** (HANDLE hMPCard, DWORD Cnt);

函数功能：设置帧重发的次数

参数说明：hMPCard：板卡的句柄

Cnt：次数设置寄存器：0 为无限次循环发送，n 为 n 次循环发送（n>0）

返回值：　若设置成功，返回值为真；否则为假

**8、 BC_SetRetryNum**

函数原型：void __stdcall **BC_SetRetryNum** (HANDLE hMPCard, BYTE Num);

函数功能：设置 BC 重试的次数

参数说明：hMPCard：板卡的句柄

Num：BC 重试次数设置　0---重试 1 次　1---重试 2 次

返回值：　无

**9、 BC_SetRetryCase**

函数原型：BOOL __stdcall **BC_SetRetryCase** (HANDLE hMPCard, RETRY_CASE_STRUCT *Retry);

函数功能：设置 BC 重试的条件

参数说明：hMPCard：板卡的句柄

Retry：指向消息重试结构指针

返回值：　若设置成功，返回值为真；否则为假

**10、 BC_RetryChanSel**

函数原型：BOOL __stdcall BC_**RetryChanSel** (HANDLE hMPCard, RETRY_CHANNEL_SEL_STRUCT *ChanSel);

函数功能：设置 BC 重试通道选择

参数说明：hMPCard：板卡的句柄

ChanSel：指向 BC 重试通道选择结构的指针

返回值：　若设置成功，返回值为真；否则为假

**11、 BC_StopOnError**

函数原型：BOOL __stdcall **BC_StopOnError** (HANDLE hMPCard, STOP_ON_ERR_STRUCT *Err);

函数功能：BC 停止消息处理

参数说明：hMPCard：板卡的句柄

Err：结构指针，当 RT 状态字出现 Message Error 时，BC 将停止消息或帧的处理

返回值：　若设置成功，返回置为真；否则为假

**12、 BC_OnStatusSet**

函数原型：BOOL __stdcall **BC_OnStatusSet** (HANDLE hMPCard, STATUS_SET_STRUCT *Status);

函数功能：当 RT STATUS_SET 置为时，BC 将停止消息或帧的处理功能设置

参数说明：hMPCard：板卡的句柄

Status：结构指针

返回值：　若设置成功，返回值为真；否则为假

**13、 BC_SendDataFrame**

函数原型：void __stdcall **BC_SendDataFrame** (HANDLE hMPCard, SFRAME_STRUCT *Frame);

函数功能：BC 发送数据帧

参数说明：hMPCard：板卡的句柄

Frame：待发送的数据帧

返回值：　无

**14、 BC_Start**

函数原型：void __stdcall BC_Start (HANDLE hMPCard);

函数功能：BC 开始处理命令。当 BC 将帧消息处理完或数据传输的过程中出错停止传输时，BC 将自动停止。

参数说明：hMPCard：板卡的句柄

返回值：　无

**15、 BC_IsMSGOver**

函数原型：BOOL __stdcall **BC_IsMSGOver** (HANDLE hMPCard);

函数功能：判断 BC 消息传输是否结束

参数说明：hMPCard：板卡的句柄

返回值：　如果消息传输结束，返回值为真；否则为假

**16、 BC_ReadMsg**

函数原型：BOOL __stdcall **BC_ReadMsg** (HANDLE hMPCard,

WORD *MsgId, RMSG_STRUCT *Msg);

函数功能：读取消息函数

参数说明：hMPCard：板卡的句柄

MsgId：存放消息序号（由于 BC 是按帧来发送消息，一个帧由若干个消息组成，消息序号是指该消息在该帧中的位置，下标从 0 开始）

Msg：指向读取到的数据消息

返回值：　如果读取到一条消息（消息发送结束），函数返回值为真；否则为假

**17、 BC_IsFrameOver**

函数原型：BOOL __stdcall **BC_IsFrameOver** (HANDLE hMPCard);

函数功能：判断 BC 帧是否传输结束

参数说明：hMPCard：板卡的句柄

返回值：　如果帧传输结束，返回值为真；否则为假

**18、 BC_ReadDataFrame**

函数原型：void __stdcall **BC_ReadDataFrame** (HANDLE hMPCard,

RFRAME_STRUCT *Frame);

函数功能：BC 读取数据帧函数

参数说明：hMPCard：板卡的句柄

Frame：指向读取到的数据帧

返回值：　无

## 19、 RT_Init

函数原型：void __stdcall **RT_Init** (HANDLE hMPCard);

函数功能：初始化远程终端

参数说明：hMPCard：板卡的句柄

返回值：　无

## 20、 RT_TxMode

函数原型：void __stdcall **RT_TxMode** (HANDLE hMPCard,

RT_TX_MODE_STRUCT *TxMode);

函数功能：设置数据发送的模式

参数说明：hMPCard：板卡的句柄

TxMode：指向数据发送模式结构的指针

返回值：　无

## 21、 RT_Select

函数原型：BOOL __stdcall **RT_Select** (HANDLE hMPCard,

DWORD RTEnable);

函数功能：RT 使能函数

参数说明：hMPCard：板卡的句柄

RTEnable：RT 使能寄存器，定义如下：

| 数据位 | 描述 |
|---|---|
| D0 | 1：地址为 0 的远程终端使能　　0：不使能 |
| D1 | 1：地址为 1 的远程终端使能　　0：不使能 |
| D2 | 1：地址为 2 的远程终端使能　　0：不使能 |
| D3 | 1：地址为 3 的远程终端使能　　0：不使能 |
| D4 | 1：地址为 4 的远程终端使能　　0：不使能 |
| D5 | 1：地址为 5 的远程终端使能　　0：不使能 |

**返回值：**

| D6 | 1：地址为 6 的远程终端使能　0：不使能 |
|---|---|
| D7 | 1：地址为 7 的远程终端使能　0：不使能 |
| D8 | 1：地址为 8 的远程终端使能　0：不使能 |
| D9 | 1：地址为 9 的远程终端使能　0：不使能 |
| D10 | 1：地址为 10 的远程终端使能　0：不使能 |
| D11 | 1：地址为 11 的远程终端使能　0：不使能 |
| D12 | 1：地址为 12 的远程终端使能　0：不使能 |
| D13 | 1：地址为 13 的远程终端使能　0：不使能 |
| D14 | 1：地址为 14 的远程终端使能　0：不使能 |
| D15 | 1：地址为 15 的远程终端使能　0：不使能 |
| D16 | 1：地址为 16 的远程终端使能　0：不使能 |
| D17 | 1：地址为 17 的远程终端使能　0：不使能 |
| D18 | 1：地址为 18 的远程终端使能　0：不使能 |
| D19 | 1：地址为 19 的远程终端使能　0：不使能 |
| D20 | 1：地址为 20 的远程终端使能　0：不使能 |
| D21 | 1：地址为 21 的远程终端使能　0：不使能 |
| D22 | 1：地址为 22 的远程终端使能　0：不使能 |
| D23 | 1：地址为 23 的远程终端使能　0：不使能 |
| D24 | 1：地址为 24 的远程终端使能　0：不使能 |
| D25 | 1：地址为 25 的远程终端使能　0：不使能 |
| D26 | 1：地址为 26 的远程终端使能　0：不使能 |
| D27 | 1：地址为 27 的远程终端使能　0：不使能 |
| D28 | 1：地址为 28 的远程终端使能　0：不使能 |
| D29 | 1：地址为 29 的远程终端使能　0：不使能 |
| D30 | 1：地址为 30 的远程终端使能　0：不使能 |
| D31 | 未定义 |

**返回值：**　若设置成功，返回值为真；否则为假

**22、 RT_ClearTTagOnSync**

函数原型：void __stdcall **RT_ClearTTagOnSync** (HANDLE hMPCard,

BOOL Enable);

函数功能：时标在接收到不带数据的同步方式指令后自动清零

参数说明：hMPCard：板卡的句柄

Enable：模式使能位　TRUE：使能该功能

返回值：　无

**23、 RT_LoadTTagOnSync**

函数原型：void __stdcall **RT_LoadTTagOnSync** (HANDLE hMPCard,

BOOL Enable);

函数功能：时标在接收到带数据的同步方式指令后，将命令中的数据加载到时标中作为初始值

参数说明：hMPCard：板卡的句柄

Enable：模式使能位　TRUE：使能该功能

返回值：　无

**24、 RT_Status_Set**

函数原型：DLL BOOL __stdcall **RT_Status_Set** (HANDLE hMPCard,

BYTE RTAddr, RT_STATUS_WORD_STRUCT *StatusWord);

函数功能：RT 的状态字置位

参数说明：hMPCard：板卡的句柄

RTAddr：RT 的地址，低 5 位有效

StatusWord：指向 RT 状态字设置结构的指针

返回值：　若设置成功，返回值为真；否则为假

**25、 RT_IllegalCmd**

函数原型：BOOL __stdcall **RT_IllegalCmd** (HANDLE hMPCard,

BOOL Enable);

函数功能：RT 非法命令表使能函数

参数说明：hMPCard：板卡的句柄

Enable：使能位　TRUE：使能非法命令表

返回值：　若设置成功，返回值为真；否则为假

**26、 RT_RevIllegalData**

函数原型：BOOL __stdcall **RT_RevIllegalData** (HANDLE hMPCard,

BOOL Enable);

函数功能：RT 非法指令接收数据使能

参数说明：hMPCard：板卡的句柄

Enable：使能位

TRUE：接收非法指令数据，将数据写入接收数据缓存，并将状态字中的消息差错位置位；对于发送数据指令 RT 只发送状态字（如果需要发送），且将状态字中的消息差错位置位，不发送数据。

FALSE：不接收非法指令数据，将状态字中的消息差错位置位

返回值：　若设置成功，返回值为真；否则为假

**27、 RT_SetIllegalCmdTable**

函数原型：void __stdcall **RT_SetIllegalCmdTable** (HANDLE hMPCard,

RT_Illegal_CMD_TABLE_STRUCT *CmdTable);

函数功能：RT 非法指令表设置

参数说明：hMPCard：板卡的句柄

CmdTable：指向 RT 非法命令表结构的指针

返回值：　无

**28、 RT_SetVectorWord**

函数原型：void __stdcall **RT_SetVectorWord** (HANDLE hMPCard,

BYTE RTAddr,

WORD VectorWord);

函数功能：设置矢量字

参数说明：hMPCard：板卡的句柄

RTAddr：RT 的地址

VectorWord：矢量字

返回值： 无

**29、 RT_SetBITWord**

函数原型：void __stdcall **RT_SetBITWord** (HANDLE hMPCard,

BYTE RTAddr,

WORD BITWord);

函数功能：设置字检字

参数说明：hMPCard：板卡的句柄

RTAddr：RT 的地址

BITWord：字检字

返回值： 无

**30、 RT_SendMSG**

函数原型：void __stdcall **RT_SendMSG** (HANDLE hMPCard,

BYTE RTAddr, BYTE SubAddr, WORD MsgLen, WORD *Msg);

函数功能：RT 发送数据函数

参数说明：hMPCard：板卡的句柄

RTAddr：待发送数据的远程终端

SubAddr：待发送数据的远程终端的子地址

MsgLen：待发送数据的长度，但缓冲模式下有效长度为 32

Msg：存放待发送的数据

返回值： 无

**31、 RT_ReadMSG_Rx**

函数原型：BOOL __stdcall **RT_ReadMSG_Rx** (HANDLE hMPCard,

RMSG_STRUCT *Msg);

函数功能：RT 读取接收到的数据消息

参数说明：hMPCard：板卡的句柄

Msg：存放接收到的数据消息

返回值： 如果接收到消息，返回值为真；否则为假

**32、 RT_ReadMSG_Tx**

函数原型：BOOL __stdcall **RT_ReadMSG_Tx** (HANDLE hMPCard,

RMSG_STRUCT *Msg);

函数功能：RT 读取已发送的数据消息

参数说明：hMPCard：板卡的句柄

Msg：存放已发送的数据消息

返回值： 如果 RT 发送了数据消息，返回值为真；否则为假

**33、 MT_Init**

函数原型：void __stdcall **MT_Init** (HANDLE hMPCard);

函数功能：初始化总线监视器

参数说明：hMPCard：板卡的句柄

返回值： 无

**34、 MT_SetCmdFilterTable**

函数原型：void __stdcall **MT_SetCmdFilterTable** (

HANDLE hMPCard,

MT_CMD_FILTER_TABLE_STRUCT *FTable);

函数功能：设置待监测的消息

参数说明：hMPCard：板卡的句柄

FTable：命令字过滤表结构指针

返回值： 无

**35、 MT_Start**

函数原型：BOOL __stdcall **MT_Start** (HANDLE hMPCard, BOOL Enable);

函数功能：MT 使能，并开始工作函数

参数说明：hMPCard：板卡的句柄

Enable：使能位　TRUE：MT 使能

返回值：　若 MT 使能成功，返回值为真；否则为假

**36、 MT_ReadMSG**

函数原型：BOOL __stdcall **MT_ReadMSG** (HANDLE hMPCard,

RMSG_STRUCT *Msg);

函数功能：MT 读取消息函数

参数说明：hMPCard：板卡的句柄

Msg：存取读到的消息

返回值：　如果读到消息，返回值为真；否则为假

**37、 TTL_DO**

函数原型：void __stdcall **TTL_DO** (HANDLE hMPCard, BYTE DOUT);

函数功能：8 路 TTL 数字量输出

参数说明：hMPCard：板卡的句柄

DOUT：数字量，定义如下表：

| 数据位 | 描　　述 |
|---|---|
| D0 | 第 0 路 TTL 数字量输出 |
| D1 | 第 1 路 TTL 数字量输出 |
| D2 | 第 2 路 TTL 数字量输出 |
| D3 | 第 3 路 TTL 数字量输出 |
| D4 | 第 4 路 TTL 数字量输出 |
| D5 | 第 5 路 TTL 数字量输出 |
| D6 | 第 6 路 TTL 数字量输出 |
| D7 | 第 7 路 TTL 数字量输出 |

返回值：　无

**38、 TTL_DOEN**

函数原型：BOOL __stdcall **TTL_DOEN** (HANDLE hMPCard, BOOL Enable);

函数功能：TTL 数字量输出使能

参数说明：hMPCard：板卡的句柄

Enable：使能位 TRUE：开始数字量输出

FALSE：高阻状态

返回值： 如果设置成功，返回值为真；否则为假

**39、 TTL_DI**

函数原型：BYTE __stdcall **TTL_DI** (HANDLE hMPCard);

函数功能：8 路 TTL 数字量输入

参数说明：hMPCard：板卡的句柄

返回值： TTL 数字量输入，定义如下表：

| 数据位 | 描　　述 |
| --- | --- |
| D0 | 第 0 路 TTL 数字量输入 |
| D1 | 第 1 路 TTL 数字量输入 |
| D2 | 第 2 路 TTL 数字量输入 |
| D3 | 第 3 路 TTL 数字量输入 |
| D4 | 第 4 路 TTL 数字量输入 |
| D5 | 第 5 路 TTL 数字量输入 |
| D6 | 第 6 路 TTL 数字量输入 |
| D7 | 第 7 路 TTL 数字量输入 |

### 3.4.3.3 ARINC429 部分

**1、 MP429_Reset**

函数原型：BOOL __stdcall **MP429_Reset**(HANDLE hMPCard);

函数功能：ARINC429 部分复位函数。将清空发送和接收 FIFO；停止接收数据；停止四路发送通道的定时发送；停止时标；清除时间标签、定时发送、标号过滤模式。处于不通讯状态，用户需重新设置波特率。

参数说明：hMPCard：板卡的句柄

返回值： 若复位成功，返回置为真；否则为假

**2、 EnableBaudrate_48K**

函数原型：void __stdcall **EnableBaudrate_48K** (HANDLE hMPCard,
BYTE ChannelNo, BOOL Enable);

函数功能：48K 波特率使能函数

参数说明：hMPCard：板卡的句柄

ChannelNo：取值为：

0—第 1，2 路接收通道和第 1 路发送通道 48K 波特率使能

1—第 2 路发送通道 48K 波特率使能

2—第 3，4 路接收通道和第 3 路发送通道 48K 波特率使能

3—第 4 路发送通道 48K 波特率使能

Enable：48K 波特率使能位。

TRUE：使能 48K 波特率

FALSE：不使能 48K 波特率，波特率由函数 SetConfigureWord 决定

返回值：无

**3、 SetConfigureWord**

函数原型：BOOL __stdcall **SetConfigureWord** (HANDLE hMPCard,
BYTE ChannelNo, WORD CFGWord);

函数功能：设置 ARINC429 配置字，包括波特率、数据位长度及发送奇偶校验设置

参数说明：hMPCard：板卡的句柄

ChannelNo：通道号，取值范围为 0~3

0---第 1，2 路接收通道和第 1 路发送通道波特率，数据位长度及发送奇偶校验设置

1---第 2 路发送通道波特率，数据位长度及发送奇偶校验设置

2---第 3，4 路接收通道和第 3 路发送通道波特率，

数据位长度及发送奇偶校验设置

3---第 4 路发送通道波特率，数据位长度及发送奇偶校验设置

CFGWord：配置字，结构见下表：

ChannelNo=0、2 时配置字寄存器：

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | RCVSEL | TXSEL | PARCK | Y2 | X2 | SDENB2 | Y1 | X1 | SDENB1 | SLFTST | PAREN | X | X | X | X |

**PAREN**：发送通道校验使能

1： bit32为校验位 0： bit32为数据位

**SLFTST**： 自检使能

1： 正常工作状态 0： 自检状态

在自检状态下，第一路发送通道的输出，在内部被连接到第一和第二接收通道。其中，数据在进入第二接收通道前被反相，以便与第一接收通道区分。 自检状态下，发送通道仍然正常输出。

**SDEN1**：第一路接收通道S/D码检测使能。

1： 使能第一路接收通道，源/目标 译码器 0：关闭第一路接收通道，源/目标 译码器

**X1**，**Y1**：第一路接收通道S/D码。

如果SDEN1=1，第一路接收通道将收到ARINC429数据的S/D与X1,Y1进行比较，相同则接纳，不相同则丢弃。其中，X1与串行数据bit9比较，Y1与串行数据bit10比较。

**SDEN2**：第二路接收通道S/D码检测使能。

1：使能第二路接收通道，源/目标 译码器 0：关闭第二路接收通道，源/目标 译码器

**X2**，**Y2**：第一路接收通道S/D码。

如果SDEN2=1，第二路接收通道将收到ARINC429数据的S/D与

X2,Y2进行比较，相同则接纳，不相同则丢弃。其中，X2与串行数据bit9比较，Y2与串行数据bit10比较。

**PARCK**：校验检测使能。

1： 将发送通道校验位取反（使用偶校验），以测试校验电路。 0：使用正常的奇校验。

**TXSEL**： 第一路发送通道波特率选择。

1：低速（12.5 K bps） 0：高速（100 K bps）

**RCVSEL**： 第一，二路接收通道波特率选择。

1：低速（12.5 K bps） 0：高速（100 K bps）

这样，在自检模式下，所有配置字的D5位设置为0，则第1路数据的发送，将连接收第1、2路数据的接收；第3路数据的发送，将连接收第3、4路数据的接收；但第2、4、接收通道数据相对于发送通道1、3的数据，将被反相（即取反）。

ChannelNo=1、3时配置字寄存器：

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | X | TXSEL | PARCK | X | X | X | X | X | X | X | PAREN | X | X | X | X |

**PAREN**：发送通道校验使能

1： bit32 为校验位 0： bit32 为数据位

**PARCK**：校验检测使能。

1： 将发送通道校验位取反（使用偶校验），以测试校验电路。 0：使用正常的奇校验。

**TXSEL**： 发送通道波特率选择。

1：低速（12.5 K bps） 0：高速（100 K bps）

返回值：若配置字设置成功，返回值为真；否则为假

接收恒定采用奇校验

**4、 SetLabelFilter**

函数原型：void __stdcall **SetLabelFilter** (HANDLE hMPCard, LabelTable_STRUCT *Label);

函数功能：设置标号过滤表，即某种标号和 S/D 码的 ARINC429 数据将被接收

参数说明：hMPCard：板卡的句柄

Label：指向标号过滤表设置结构的指针

返回值： 无

**5、 StartLabelFilter**

函数原型：BOOL __stdcall **StartLabelFilter** (HANDLE hMPCard,

BYTE ChannelNo, BOOL Enable);

函数功能：接收通道 ARINC429 数据的标号和 S/D 码参与过滤使能函数

参数说明：hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

Enable：使能位；

TRUE：接收通道将只接收标号和 S/D 码符合标号过滤表中的 ARINC429 数据

FALSE：接收通道将接收所有标号和 S/D 码的 ARINC429 数据

返回值：若标号过滤功能使能成功，返回值为真；否则为假

**6、 AddTimeTag**

函数原型：BOOL __stdcall **AddTimeTag** (HANDLE hMPCard, BYTE ChannelNo, BOOL Enable);

函数功能：设置时间标签模式

参数说明：hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

Enable：模式使能位；

TRUE：使能，接收到的 ARINC429 数据将带一个时间标签，表明数据接收到的时间

FALSE：不使能

返回值：若时标模式设置成功，返回值为真；否则为假

**7、 StartTimeTag**

**函数原型：**void __stdcall **StartTimeTag** (HANDLE hMPCard, BOOL Enable, LPSYSTEMTIME CurSysTime);

**函数功能：**接收方式下时标开始计数函数，否则接收的时标为 0

**参数说明：**hMPCard：板卡的句柄

Enable：使能位；TRUE：使能，时标开始计数

FALSE：不使能，时标停止计数

CurSysTime：系统当前时间

```
typedef struct _SYSTEMTIME {
    WORD wYear;
    WORD wMonth;
    WORD wDayOfWeek;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
    WORD wMilliseconds;
} SYSTEMTIME, *LPSYSTEMTIME;
```

**wYear** ：年

**WMonth** ：月; January = 1, February =2，以此类推

**wDayOfWeek**：星期; Sunday = 0, Monday = 1，以此类推

**wDay** ：日.

**wHour** ：小时

**wMinute** ：分.

**wSecond** ：秒

**wMilliseconds** ：毫秒

**返回值：**若时标使能成功，返回值为真；否则为假

**8、 SetTimerInterval**

函数原型：BOOL __stdcall **SetTimerInterval**(HANDLE hMPCard,

BYTE ChannelNo, WORD Time);

函数功能：设置定时发送时间间隔。定时发送的时间间隔为两次启动定时发送数据的起始时间的时间差；定时发送的时间应该保证发送完所有的数据，否则发送的数据会不全。

参数说明：hMPCard：板卡的句柄

ChannelNo：通道号，取值范围为 0~3，对应板卡的发送 1~4 路

Time：定时时间，单位为 50 微秒

返回值：若定时时间设置成功，返回值为真；否则为假

**9、 StartTimer**

函数原型：BOOL __stdcall **StartTimer**(HANDLE hMPCard, BYTE ChannelNo,

BOOL Enable);

函数功能：启动或停止定时发送

参数说明：hMPCard：板卡的句柄

ChannelNo：通道号，取值为 0~3，对应板卡的发送 1~4 路

Enable：使能位；

TRUE：若设置了定时发送时间，硬件将按定时时间间隔重复发送 FIFO 里的数据

FALSE：硬件将停止数据的发送

返回值：若定时器启动成功，返回值为真；否则为假

**10、 SetTriggerDepth**

函数原型：void __stdcall **SetTriggerDepth** (HANDLE hMPCard,

TriggerDepth_STRUCT *Depth);

函数功能：设置 FIFO 触发深度

参数说明：hMPCard：板卡的句柄

Depth：FIFO 触发深度结构指针

返回值：空

**11、 MP429_CheckInit**

函数原型：BOOL **__**stdcall **MP429_CheckInit**(HANDLE hMPCard,

BOOL isCheck);

函数功能：ARINC429 查询接收初始化

参数说明：hMPCard：板卡的句柄

isCheck：是否进行查询接收

TRUE：进行查询接收

FALSE：不进行查询接收初始化，程序采用中断接收

返回值：若操作成功，返回真，否则返回假

**12、 CountsInRevFIFO**

函数原型：DWORD **__**stdcall **CountsInRevFIFO**(HANDLE hMPCard,

BYTE ChannelNo);

函数功能：读取接收 FIFO 中的数据量（每一路接收通道具有一个接收缓存，用来存放接收到的数据）

参数说明：hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

返回值： FIFO 中 ARINC429 数据的个数

**13、 IsFIFOTriggered_R**

函数原型：BOOL **__**stdcall **IsFIFOTriggered_R** (HANDLE hMPCard,

BYTE ChannelNo);

函数功能：判断接收 FIFO 是否到达触发深度（每一路接收通道具有一个接收缓存，用来存放接收到的数据）

参数说明：hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

返回值： 若接收 FIFO 达到触发深度，返回值为真；否则为假

**14、 IsFIFOTriggered_S**

**函数原型：**BOOL __stdcall **IsFIFOTriggered_S** (HANDLE hMPCard,

BYTE ChannelNo);

**函数功能：**判断发送 FIFO 是否到达触发深度（每一路发送通道带一个发送 FIFO，用来存放待发送的数据）

**参数说明：**hMPCard：板卡的句柄

ChannelNo：发送通道号，取值范围为 0~3，分别对应发送通道 1~4

**返回值：** 若 FIFO 到达触发深度，返回值为真；否则为假

**15、 ReadFIFOStatus_R**

**函数原型：**BYTE __stdcall **ReadFIFOStatus_R** (HANDLE hMPCard,

BYTE ChannelNo);

**函数功能：**在查询接收时读取接收 FIFO 的状态，包括接收 FIFO 的空、满、溢出标志

**参数说明：**hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

**返回值：** 返回值见下表：

| 二进制值 | 符号名 | 意义 |
|---|---|---|
| 0x10 | FIFOEmpty | FIFO 空 |
| 0x11 | FIFONotFull | FIFO 不满 ，但也不空 |
| 0x12 | FIFOError | FIFO 溢出 |
| 0x13 | ErrIndex | 错误通道号 |
| 0x14 | FIFOFull | FIFO 满 |

**16、 ReadFIFOStatus_S**

**函数原型：**BYTE __stdcall **ReadFIFOStatus_S** (HANDLE hMPCard,

BYTE ChannelNo);

**函数功能：**读取发送 FIFO 的状态，包括发送 FIFO 的空、满、溢出标志

**参数说明：**hMPCard：板卡的句柄

ChannelNo：发送通道号，取值范围为 0~3，分别对应发送通道 1~4

**返回值：** 返回值见下表

| 二进制值 | 符号名 | 意义 |
| --- | --- | --- |
| 0x10 | FIFOEmpty | FIFO 空 |
| 0x11 | FIFONotFull | FIFO 不满 ，但也不空 |
| 0x12 | FIFOError | FIFO 溢出 |
| 0x13 | ErrIndex | 错误通道号 |
| 0x14 | FIFOFull | FIFO 满 |

**17、 FIFOStatus**

**函数原型：** BYTE __stdcall **FIFOStatus** (HANDLE hMPCard, BYTE ChannelNo);

**函数功能：** 采用中断接收时读取接收 FIFO 的状态，包括发送 FIFO 的空、满、溢出标志

**参数说明：** hMPCard：板卡的句柄

ChannelNo：发送通道号，取值范围为 0~3，分别对应发送通道 1~4

**返回值：** 返回值见下表

| 二进制值 | 符号名 | 意义 |
| --- | --- | --- |
| 0x10 | FIFOEmpty | FIFO 空 |
| 0x11 | FIFONotFull | FIFO 不满 ，但也不空 |
| 0x12 | FIFOError | FIFO 溢出 |
| 0x13 | ErrIndex | 错误通道号 |
| 0x14 | FIFOFull | FIFO 满 |

**18、 ReceiveData_Dword**

**函数原型：** DWORD __stdcall **ReceiveData_Dword** (HANDLE hMPCard, BYTE ChannelNo);

**函数功能：** 接收数据函数。用于在查询接收时接收一个 ARINC429 数据（32bits）

**参数说明：** hMPCard：板卡的句柄

ChannelNo：待接收数据的通道号，取值范围为 0~3，分别对应接

收 1~4 路

返回值：接收到的 32 位数据

**19、 ReceiveData_CBlock**

函数原型：void __stdcall **ReceiveData_CBlock** (HANDLE hMPCard, BYTE ChannelNo，WORD Len, DWORD* Data);

函数功能：接收数据块函数,用于在查询接收时接收指定长度的数据

参数说明：hMPCard：板卡的句柄

ChannelNo：待接收数据的通道号，取值范围为 0~3，分别对应接收 1~4 路

Len：接收数据量的个数，范围为小于等于所设置的接收触发深度

Data：存放接收数据的指针

返回值：无

**20、 ReceiveData_IBlock**

函数原型：void __stdcall **ReceiveData_IBlock** (HANDLE hMPCard, BYTE ChannelNo，WORD Len, DWORD* Data);

函数功能：接收数据块函数,用于在中断接收时接收指定长度的数据

参数说明：hMPCard：板卡的句柄

ChannelNo：待接收数据的通道号，取值范围为 0~3，分别对应接收 1~4 路

Len：接收数据量的个数，范围为小于等于所设置的接收触发深度

Data：存放接收数据的指针

返回值：无

**21、 SendData**

函数原型：BOOL __stdcall **SendData** (HANDLE hMPCard, BYTE ChannelNo, BYTE Len, DWORD *DataAry);

函数功能：数据发送函数

参数说明：hMPCard：板卡的句柄

ChannelNo：待发送数据的通道号，取值范围为 0~3，分别对应发送 1~4 通道

Len：待发送数据的长度，最大为 255，单位：32-bit

DataAry：存放待发送的数据

返回值：若数据发送成功，返回值为真；否则为假

**22、 Card_IntCreate**

函数原型：BOOL **Card_IntCreate** (HANDLE hMPCard, INT_HANDLER funcIntHandler);

函数功能：创建板卡中断

参数说明：hMPCard：板卡的句柄

funcIntHandler：中断服务函数名

返回值：若操作成功，返回值为真；否则为假

**23、 IntEnable**

函数原型：BOOL IntEnable(HANDLE hMPCard,BOOL Enable);

函数功能：中断使能

参数说明：hMPCard：板卡的句柄

Enable：使能位；

TRUE：中断使能

FALSE：中断禁止

返回值：使能成功，返回值为真；否则为假

**24、 EnTriggerInt_R**

函数原型：BOOL **EnTriggerInt_R**(HANDLE hMPCard,BYTE ChannelNo, BOOL Enable);

函数功能：使能接收数据触发中断函数

参数说明：hMPCard：板卡的句柄

ChannelNo：待接收数据的通道号，取值范围为 0~3；分别对应接收通道 1~4

Enable：使能位；

TRUE：接收触发中断使能

FALSE：接收触发中断禁止

**返回值**：使能成功，返回值为真；否则为假

**25、 EnTriggerInt_S**

**函数原型**：BOOL **EnTriggerInt_S**(HANDLE hMPCard,BYTE ChannelNo, BOOL Enable);

**函数功能**：单路使能发送触发中断函数

**参数说明**：hMPCard：板卡的句柄

ChannelNo：待接收数据的通道号，取值范围为 0~3；分别对应接收通道 1~4

Enable：使能位；

TRUE：发送触发中断使能

FALSE：发送触发中断禁止

**返回值**：使能成功，返回值为真；否则为假

**26、 EnTriggerInt_SALL**

**函数原型**：BOOL **EnTriggerInt_SALL**(HANDLE hMPCard,BOOL Enable);

**函数功能**：同时使能四路发送触发中断函数

**参数说明**：hMPCard：板卡的句柄

Enable：使能位；

TRUE：发送触发中断使能

FALSE：发送触发中断禁止

**返回值**：使能成功，返回值为真；否则为假

**27、 IntStatus**

**函数原型**：DWORD __stdcall **IntStatus** (HANDLE hMPCard);

**函数功能**：返回中断状态寄存器的值

**参数说明**：hMPCard：板卡的句柄

**返回值**：中断状态寄存器的值，由此判断是何种条件引发了中断，定义如下：

| BITS | DESCRIPTION | |
|---|---|---|
| D0 | 1：接收第 1 路产生中断 | 0：接收第 1 路无中断 |
| D1 | 1：接收第 2 路产生中断 | 0：接收第 2 路无中断 |
| D2 | 1：接收第 3 路产生中断 | 0：接收第 3 路无中断 |
| D3 | 1：接收第 4 路产生中断 | 0：接收第 4 路无中断 |
| D4 | 1：发送第 1 路产生中断 | 0：发送第 1 路无中断 |
| D5 | 1：发送第 2 路产生中断 | 0：发送第 2 路无中断 |
| D6 | 1：发送第 3 路产生中断 | 0：发送第 3 路无中断 |
| D7 | 1：发送第 4 路产生中断 | 0：发送第 4 路无中断 |
| D8 | 1：产生中断 | 0：无中断 |
| D9-D31 | 保留 | |

### 3.4.3.4 串口部分

**1、 MPRS_Reset**

**函数原型**：BOOL __stdcall **MPRS_Reset**(HANDLE hMPCard);

**函数功能**：串口复位函数，板卡将清空发送和接收 FIFO，此时板卡为 RS232 模式， 发送通道无校验、1 位停止位、8 位数据位，接收通道为无校验、8 位数据位、透明接收，此时串口不工作，需重新设置发送波特率和接收波特率。

**参数说明**：hMPCard：板卡的句柄

**返回值**： 若复位成功，返回置为真；否则为假

**2、 RS_ProtocolSet**

**函数原型**：void __stdcall **RS_ProtocolSet** (HANDLE hMPCard, BYTE ChannelNo, BYTE Protocol);

**函数功能**：串口传输协议设置函数

**参数说明**：hMPCard：板卡的句柄

ChannelNo：串口通道号，取值范围为 0~3；分别对应串口通道 1~4

Protocol：传输协议选择，取值如下所示：

| 二进制值 | 符号名 | 意义 |
|---|---|---|
| 0x00 | RS232 | RS232 模式 |
| 0x01 | RS422 | RS422/ RS485 模式 |

返回值：　无

**3、 RS_CommunModeSet**

函数原型：void __stdcall **RS_CommunModeSet** (HANDLE hMPCard,

BYTE ChannelNo, BYTE CommunMode);

函数功能：RS422/RS485 传输协议下通信模式设置函数

参数说明：hMPCard：板卡的句柄

ChannelNo：串口通道号，取值范围为 0~3；分别对应串口通道 1~4

CommunMode：通信模式设置，取值如下所示：

| 二进制值 | 符号名 | 意义 |
|---|---|---|
| 0x00 | FULL_DUPLEX | 全双工通信 |
| 0x01 | HALF_DUPLEX | 半双工通信 |

返回值：　无

**4、 RS_TranDirSet**

函数原型：void __stdcall RS_TranDirSet(HANDLE hMPCard, BYTE ChannelNo,

BYTE HDDirection);

函数功能：RS422/RS485 传输协议下半双工模式下传输方向设置函数

参数说明：hMPCard：板卡的句柄

ChannelNo：串口通道号，取值范围为 0~3；分别对应串口通道 1~4

HDDirection：传输方向设置，取值如下所示：

| 二进制值 | 符号名 | 意义 |
|---|---|---|
| 0x00 | HDREVDIR | 接收 |
| 0x01 | HDSENDDIR | 发送 |

返回值：　无

**5、 RS_ProFrameSet**

函数原型：void __stdcall **RS_ProFrameSet**(HANDLE hMPCard,

BYTE ChannelNo, BYTE FrameHead, BYTE FrameLength);

函数功能：协议接收时帧格式设置函数

参数说明：hMPCard：板卡的句柄

ChannelNo：串口通道号，取值范围为 0~3；分别对应串口通道 1~4

FrameHead：帧头，8bits

FrameLength：帧长度，8 位

返回值：　无

**6、 RS_SendConfigSet**

函数原型：void __stdcall RS_SendConfigSet(HANDLE hMPCard, BYTE

ChannelNo, DWORD SBaud, BYTE SBits,BYTE SParity,BYTE SStop);

函数功能：发送通道配置字设置函数

参数说明：hMPCard：板卡的句柄

ChannelNo：串口通道号，取值范围为 0~3；分别对应串口通道 1~4

SBaud：波特率

SBits：数据位长度　0x00：8 位，0x01：7 位，

0x02：6 位，0x03：5 位

SParity：奇偶校验位 0x00：无校验，0x01：奇校验，0x02：偶校验

SStop：停止位　0：1 位，1：2 位

返回值：　无

**7、 RS_ RevConfigSet**

函数原型：void __stdcall RS_RevConfigSet(HANDLE hMPCard, BYTE

ChannelNo, BYTE RevMode, DWORD RBaud, BYTE RBits, BYTE RParity);

函数功能：发送通道配置字设置函数

参数说明：hMPCard：板卡的句柄

ChannelNo：串口通道号，取值范围为 0~3；分别对应串口通道 1~4

RevMode：接收模式，取值如下所示：

| 二进制值 | 符号名 | 意义 |
| --- | --- | --- |
| 0x00 | CLARITY | 透明接收 |
| 0x01 | PROTOCOL | 协议接收 |

RBaud：波特率

RBits：数据位长度　0x00：8 位，0x01：7 位，

0x02：6 位，0x03：5 位

RParity：奇偶校验位 0x00：无校验，0x01：奇校验，0x02：偶校验

**返回值：**　无

**8、 RS_ReadFIFOStatusR**

**函数原型：**BYTE __stdcall **RS_ReadFIFOStatusR** (HANDLE hMPCard,

BYTE ChannelNo);

**函数功能：**读取接收 FIFO 的状态，包括接收 FIFO 的空、满标志

**参数说明：**hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

**返回值：**　返回值见下表：

| 二进制值 | 符号名 | 意义 |
| --- | --- | --- |
| 0x10 | FIFOEmpty | FIFO 空 |
| 0x11 | FIFONotFull | FIFO 不满 ，但也不空 |
| 0x13 | ErrIndex | 错误通道号 |
| 0x14 | FIFOFull | FIFO 满 |

**9、 RS_ReadFIFOStatusS**

**函数原型：**BYTE __stdcall **RS_ReadFIFOStatusS** (HANDLE hMPCard,

BYTE ChannelNo);

**函数功能：**读取发送 FIFO 的状态，包括发送 FIFO 的空、满标志

**参数说明：**hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

**返回值：** 返回值见下表：

| 二进制值 | 符号名 | 意义 |
|---|---|---|
| 0x10 | FIFOEmpty | FIFO 空 |
| 0x11 | FIFONotFull | FIFO 不满 ，但也不空 |
| 0x13 | ErrIndex | 错误通道号 |
| 0x14 | FIFOFull | FIFO 满 |

**10、 RS_ ReceiveData**

**函数原型：** WORD __stdcall RS_ReceiveData(HANDLE hMPCard,

BYTE ChannelNo);

**函数功能：** 读取接收数据

**参数说明：** hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

**返回值：** 接收到的串口数据，16bits

**11、 RS_ SendData**

**函数原型：** WORD __stdcall RS_SendData(HANDLE hMPCard,BYTE ChannelNo ,

WORD Len, BYTE *DataAry);

**函数功能：** 发送串口数据

**参数说明：** hMPCard：板卡的句柄

ChannelNo：接收通道号，取值范围为 0~3，分别对应接收通道 1~4

Len：待发送数据个数，取值为 0~2047，单位为 8bits

DataAry：存放待发送数据的指针

**返回值：** 接收到的串口数据，16bits

## 3.5. 驱动函数调用步骤

### 3.5.1 打开板卡

调用函数 MPCard_Open 打开板卡

### 3.5.2 1553 部分

#### 3.5.2.1 复位 1553

调用函数 MP1553_Reset 复位 1553 部分

#### 3.5.2.2 初始化板卡

1、 启动或停止时间标签模式（MP1553_AddTimeTag）

2、 设置应答超时（MP1553_SetResponseTimeout）

#### 3.5.2.3 BC 模式

1、BC 初始化（BC_Init）

2、BC 帧设置

I、 设置帧间间隔（BC_SetFrameGap）

II、 启动或停止帧重复发送模式（BC_FrameAutoRepeat）

III、若设置了帧重复发送模式，则设置帧重复发送的次数（BC_FrameAuto- Repeat_Count）

3、BC 重试

I、 BC 重试的次数设置（BC_SetRetryNum）

II、 BC 重试的条件设置（BC_SetRetryCase）

III、BC 重试通道选择（BC_RetryChanSel）

4、消息出错时，BC 停止消息处理（BC_StopOnError）

5、RT STATUS_SET 置位时，BC 消息处理（BC_OnStatusSet）

6、发送数据帧

I、 写待发送的数据帧（BC_SendDataFrame）

II、 启动 BC（BC_Start）

III、 重复 I~II 步

7、读取数据帧

I、 按消息读取数据

a、判断消息传输是否结束（BC_IsMSGOver）

b、若消息结束，则读取消息（BC_ReadMsg）

c、重复 a~b 两步

II、 按帧读区数据

a、判断帧传输是否结束（BC_IsFrameOver）

b、 若帧传输结束，则读取数据帧（BC_ReadDataFrame）

c、重复 a~b 两步

### 3.5.2.4 RT 模式

1、RT 初始化（RT_Init）

2、设置 RT 数据发送模式（RT_TxMode）

3、清置时标

I、 清除时标（RT_ClearTTagOnSync）

II、 重新装载时标（RT_LoadTTagOnSync）

4、RT 状态字设置（RT_Status_Set）

5、RT 非法命令设置

I、 启动或停止 RT 非法命令表模式（RT_IllegalCmd）

II、 若启动了 RT 非法命令表模式，则设置是否接收非法命令的数据（RT_RevIllegalData）及 RT 非法命令表（RT_SetIllegalCmdTable）

6、 设置矢量字（RT_SetVectorWord）

7、 设置自检字（RT_SetBITWord）

8、 RT 使能（RT_Select）

9、 发送数据消息（RT_SendMSG）

10、接收数据消息

I、 接收已发送的数据消息（RT_ReadMSG_Tx）

II、 接收已接收到的数据消息（RT_ReadMSG_Rx）

### 3.5.2.5 MT 模式

1、初始化 MT（MT_Init）

2、设置 MT 待监控的消息（MT_SetCmdFilterTable）

3、MT 使能（MT_Start）

4、读取 MT 消息（MT_ReadMSG）

#### 3.5.2.6 TTL 数字量 IO

1、 TTL 数字量输出（TTL_DO）

2、 使能数字量输出（TTL_DOEN）

3、 TTL 数字量输入（TTL_DI）

### 3.5.3 ARINC429 部分

#### 3.5.3.1 复位 ARINC429

调用函数 MP429_Reset 复位 ARINC429 部分

#### 3.5.3.2 设置数据格式

1、 设置配置字（SetConfigureWord）

2、 启动或停止 48K 波特率（EnableBaudrate_48K）

#### 3.5.3.3 设置 FIFO 触发深度

调用函数 SetTriggerDepth 设置 FIFO 触发深度

#### 3.5.3.4 标号过滤功能的设置

1、 设置待接收数据的标号（SetLabelFilter）

2、 启动标号过滤功能（StartLabelFilter）

#### 3.5.3.5 时间标签功能设置

1、 设置时间标签模式（AddTimeTag）

2、 启动时标（StartTimeTag）

#### 3.5.3.6 定时发送功能设置

1、 设置定时发送的时间间隔（SetTimerInterval）

2、 启动定时发送模式（StartTimer）

#### 3.5.3.7 数据的接收

1、 查询方式

- 按触发深度的方式读取数据
  - a) 判断接收缓存是否触发（IsFIFOTriggered_R）
  - b) 若 FIFO 触发，则读取触发深度+1 个数据（ReceiveData_CBlock），**注：当添加时标时，必须保证读取的数据个数为偶数**
  - c) 重复以上 2 步
- FIFO 不空就读取数据
  1、 判断接收 FIFO 是否为空（ReadFIFOStatus_R）
  2、 若 FIFO 不空，读取数据（ReceiveData_Dword）
  3、 重复以上 2 步

2、 中断方式

- a) 创建中断（Card_IntCreate）
- b) 使能中断（IntEnable）
- c) 使能接收触发中断（EnTriggerInt_R）
- d) 创建中断服务程序，在其中执行如下操作：
  - ➢ 读取中断状态（IntStatus）
  - ➢ 当达到触发深度时，接收触发深度+1 个数据（ReceiveData_IBlock）
    **注：当添加时标时，必须保证读取的数据个数为偶数**
  - ➢ 开启中断（IntEnable），在中断响应时，关闭了中断

**说明**：若设置了时间标签模式，则接收数据时，首先收到的是一个 ARINC429 数据（记为 data429），接下来的是该 ARINC429 数据的时标（记为 TimeTag）。**data429 接收到的时间=时标开始计数时的系统时间+TimeTag*时标分辨率，**时标开始计数系统时间可由函数 StartTimeTag 获得。

### 3.5.3.8 数据的发送

1、 普通发送

- a) 读取发送 FIFO 的状态（ReadFIFOStatus_S）

b) 若发送 FIFO 不满或为空（为 FIFONotFull 或 FIFOEmpty），则发送数据（SendData）

2、 定时发送

a) 停止定时器（StartTimer），此时将清空发送缓冲区

b) 启动定时器（StartTimer）

c) 读取发送 FIFO 的状态（ReadFIFOStatus_S），若发送 FIFO 不满或为空（为 FIFONotFull 或 FIFOEmpty），则发送数据（SendData）

3、 触发发送

a) 创建中断（Card_IntCreate）

b) 使能中断（IntEnable）

c) 使能触发发送中断（EnTriggerInt_S 或 EnTriggerInt_SALL(Demo 程序中使用)）

d) 创建中断服务程序，在其中执行如下操作：

- 读取中断状态（IntStatus）
- 若达到触发深度，则读取发送 FIFO 的状态（ReadFIFOStatus_S）
- 若发送 FIFO 不满或为空（为 FIFONotFull 或 FIFOEmpty），则发送数据（SendData）
- 开启中断

## 3.5.4 串口部分

### 3.5.4.1 复位串口

调用函数 MPRS_Reset 复位串口。

### 3.5.4.2 设置传输协议

调用函数 RS_ProtocolSet 设置传输协议，若传输协议为 RS422/RS485，则调用函数 RS_CommunModeSet 设置通信模式。若传输协议为 RS422/RS485 且通信模式为半双工，则调用函数 RS_TranDirSet 设置传输方向为接收或发送。

### 3.5.4.3 设置发送配置字

调用函数 RS_SendConfigSet 设置发送配置字

#### 3.5.4.4 设置接收配置字

调用函数 RS_RevConfigSet 设置接收配置字，若采用协议接收，则调用函数 RS_ProFrameSet 设置接收帧格式。

#### 3.5.4.5 数据的发送

1、读取发送 FIFO 的状态（RS_ReadFIFOStatusS）

2、若发送 FIFO 不满或为空（为 FIFONotFull 或 FIFOEmpty），则发送数据（RS_SendData）

#### 3.5.4.6 数据的接收

1、判断接收 FIFO 是否为空（RS_ReadFIFOStatusR）

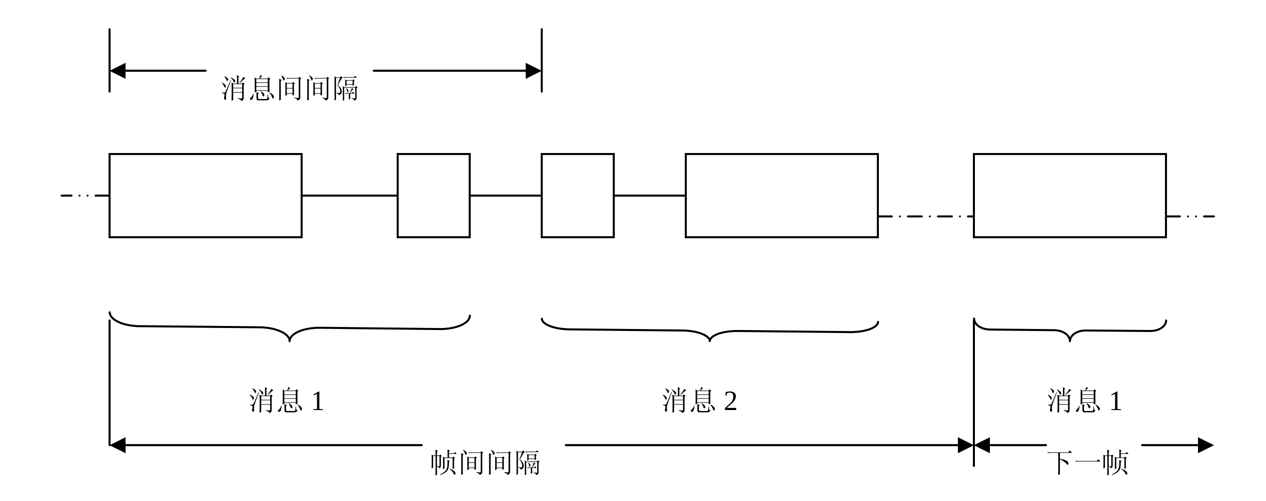
2、若接收 FIFO 不为空（即返回值不为 FIFOEmpty），则读取数据（RS_ReceiveData）

3、重复以上 2 步

### 3.5.5 关闭板卡

应用程序退出时，调用函数 MPCard_Close 关闭板卡

## 附：BC 消息间间隔和帧间间隔



# 第四章　应用程序说明

## 4.1 软件概述

MPApp 应用程序是本公司针对多协议板卡产品开发的应用程序，目的是为了使用户很快地对硬件的功能有所了解。本程序现仅供用户做产品参考与应用演示使用。

在演示程序中使用了 microsoft FlexGrid Control，如果机器上未安装该控件，程序将不能正常运行。在附带光盘“应用程序\extra”目录下含有该控件，先用 regsvr32.exe 注册此控件（msflxgrd.ocx），再将.reg 文件导入注册表。

## 4.2 软件运行环境

### 4.2.1 硬件

可支持 PCI 总线的计算机

至少 128M 内存

显示器分辨率至少可设为 800*600

AAMC-PCI-MP 通讯板卡

### 4.2.2 支持软件

Windows 98/2000/Xp 操作系统

AAMC-PCI-MP 硬件驱动程序

### 4.2.3 软件运行

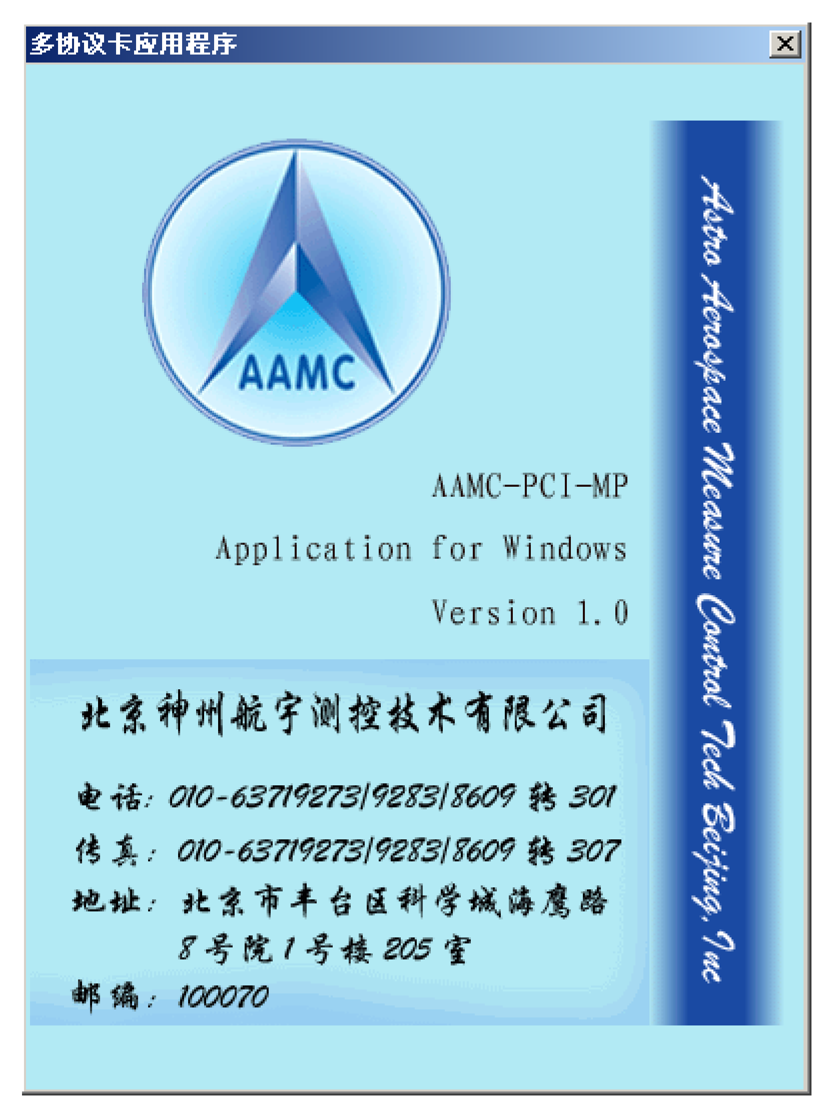
在我公司提供给用户的光盘中，用户可以找到 MPApp 的应用程序源代码，重新编译后即可运行。

在运行前建议用户要确认目标设备（即用户所使用的 PC 设备）中已经插入了我公司提供了多协议通讯板，并确保软件工作环境均符合 4.2 中的要求，最后确认是否将我公司提供的多协议通讯板驱动程序正确地安装到您的系统中。

## 4.3 软件简介

### 4.3.1 软件主体界面

图 I-1（MPApp 应用程序初始化界面）：




应用程序初始化窗口会在程序初始化时显示，程序初始化完成后 3 秒钟后，初始化窗口会自动消失，或是用鼠标点击该窗口中任意位置，窗口也会关闭。

图 I-2（板卡号选择窗口）:

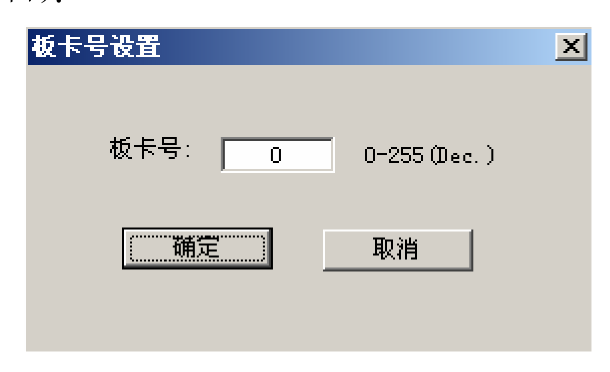
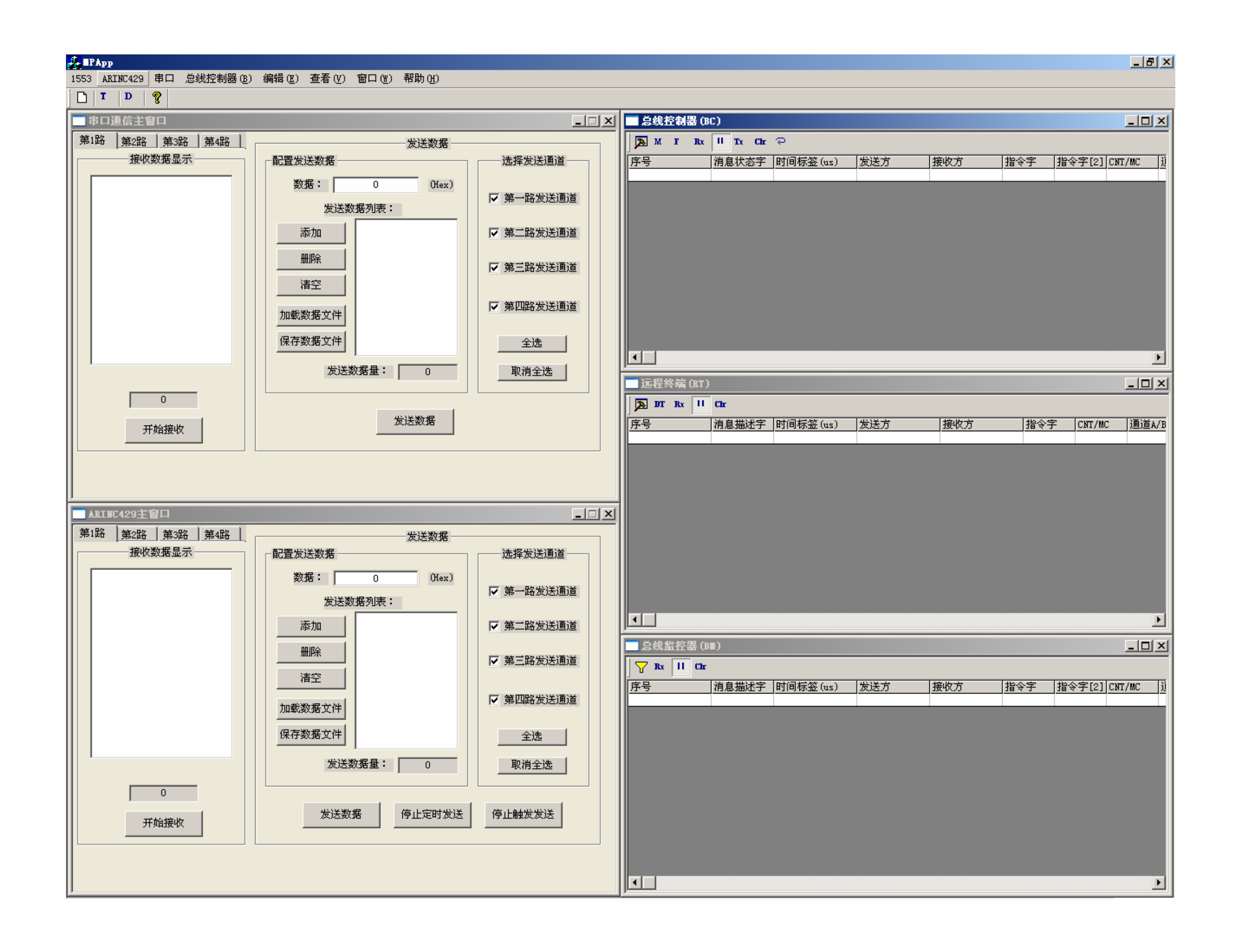



图 I-3（MPApp 应用程序主窗口）:



## 4.3.2 菜单功能

### 4.3.2.1553 菜单命令

1553 菜单提供了以下命令：

| | |
|---|---|
| 新建模式 | 创建 BC、RT、BM 操作(工作模式)。 |
| 打开数据文件 | 打开 BC、RT、BM 数据文件。 |
| 保存数据文件 | 将操作活动窗口中的数据存储到文件中，当前如果没有操作模式窗口打开时，该功能无效。 |
| 设置时标与超时 | |
| 数字量 I/O | |

注：

BC --- 总线控制器：Bus Controller

RT --- 远程终端：Remote Terminal

BM --- 总线监控器：Bus Monitor

#### 4.3.2.2 总线控制器（BC）菜单命令

总线控制器（BC）：Bus Controller。

定义：总线控制器是总线系统中组织信息传输的终端。

总线控制器命令如下:

| 设置 | 显示设置窗口。 |
|---|---|
| 消息 | 显示消息配置窗口。 |
| 创建消息帧 | 显示帧配置窗口。 |
| 开启接收 | 开启接收操作。 |
| 停止接收 | 停止接收操作。 |
| 发送数据帧 | 将用户配置好的数据帧写入硬件。 |
| 自动重发数据帧 | 开启/停止 数据帧的“自动重发”功能。 |
| 清空接收数据 | 清空接收数据显示列表。 |

注：

操作菜单中的“设置”、“消息”、“创建消息帧”与“清空接收数据”时，需要处于停止接收状态。

#### 4.3.2.3 远程终端（RT）菜单命令

远程终端（RT）：Remote Terminal。

定义：远程终端是总线系统中不作为总线控制器或总线监控器的所有终端。

远程终端菜单命令如下:

| 设置 | 显示设置窗口。 |
|---|---|
| 数据配置 | 显示数据配置窗口。 |
| 开启接收 | 开启接收操作。 |
| 停止接收 | 停止接收操作。 |
| 清空接收数据 | 清空接收数据显示列表。 |

注：

操作菜单中的“设置”、与“清空接收数据”时，需要处于停止接收状态。

#### 4.3.2.4 总线监控器（BM）菜单命令

总线监控器（BM）：Bus Monitor。

定义：总线监控器是总线系统中指定作接收且记录总线上传输的信息并有选择地提取信息以备后用的终端。

总线监控器菜单命令如下:

| 过滤器 | 显示过滤设置窗口。 |
| --- | --- |
| 开启接收 | 开启接收操作。 |
| 停止接收 | 停止接收操作。 |
| 清空接收数据 | 清空接收数据显示列表。 |

注：

操作菜单中的“过滤器”、与“清空接收数据”时，需要处于停止接收状态。

#### 4.3.2.5 ARINC429 菜单命令

点击弹出 ARINC429 通讯主窗口。

#### 4.3.2.6 ARINC429 设置菜单命令

点击弹出 ARINC429 设置窗口，用来设置 ARINC429 工作的参数。

#### 4.3.2.7 串口菜单命令

点击弹出串口通讯主窗口。

#### 4.3.2.8 串口设置菜单命令

点击弹出串口设置窗口，用来设置串口工作的参数。

#### 4.3.2.9 查看菜单命令

查看菜单提供了以下命令：

| 工具栏 | 显示或隐藏工具栏。 |
| --- | --- |
| 状态栏 | 显示或隐藏状态栏。 |

#### 4.3.2.10 窗口菜单命令

窗口菜单提供了以下命令。这些命令使您能在应用程序窗口中安排多个文档的多个视图：

| 层叠 | 按重叠方式安排窗口。 |
|---|---|
| 平铺 | 按互不重叠平铺方式安排窗口。 |
| 活动窗口 1，2，... | 转到指定的窗口 |

#### 4.3.2.11 帮助菜单命令

帮助菜单提供以下的命令，为您提供使用这个应用程序的帮助：

帮助主题　　　　提供您可从其得到帮助的主题索引。

关于　　　　　　显示这个应用程序的版本号。

### 4.3.3 软件操作建议

#### 4.3.3.1 启动应用程序

#### 4.3.3.2 板卡号选择

当在同一设备中插入多块同样的板卡时，应用程序要求指定板卡号，板卡号选择范围为 0 到 255，默认为 0 号。

#### 4.3.3.3 1553 部分

**1、 设置时间标签与应答超时**

启动或停止时间标签，并设置应答超时时间。

**2、 新建工作模式**

执行菜单中的“1553-〉新建工作模式”命令，从“新建工作模式窗口”中选择需要创建的工作模式。

**3、 总线控制器（BC）操作**

设置总线控制器（BC）；

配置消息；

创建消息帧；

开启接收消息；

发送数据帧；

停止接收消息。

**4、 远程终端（RT）操作**

设置远程终端（RT）；

配置 RT 的数据；

开启接收消息；

停止接收消息。

**5、 总线监控器（BM）操作**

消息过滤设置；

开启监视消息；

停止监视消息。

### 4.3.3.4 ARINC429 部分

**1、 ARINC429 通讯主窗口**

执行菜单中的“ARINC429”可弹出 ARINC429 通讯主窗口，在进入主窗口后，程序会自动对板卡进行默认配置，该配置结果允许用户可以进行最基本的数据发送与接收操作，即无触发深度、无定时发送、无时间标签，也无标号过滤。要进行更多的功能配置，还需要进一步设置板卡。

**2、 ARINC429 设置窗口**

执行菜单“ARINC429 设置”，程序会显示 ARINC429 设置窗口，这个窗口中包含了硬件所有可被设置的属性，如下所示：

波特率

校验位

定时时间

添加时间标签

触发深度

标号过滤

ARINC429 工作模式设置

完成设置后，单击“确定”按钮向板卡提交设置，单击“取消”结束本次操作。

#### 4.3.3.3 串口部分

**1、 串口通讯主窗口**

执行菜单中的“串口”可弹出串口通讯主窗口，可进行基本的发送和接收操作。

**2、 串口设置窗口**

传输协议

通信模式

传输方向

发送通道波特率、校验位、停止位、数据位长度

接收通道波特率、校验位、数据位长度、接收模式

协议接收时接收帧格式的设置

## 4.4 软件使用详细说明

MPApp 应用程序包括三种协议：1553、ARINC429 和串口，其中 1553 包括三种工作模式：总线控制器（BC）、 远程终端（RT）、总线监控器 （BM）。

如图 I-3 所示，MPApp 应用程序窗口为多文本窗口形式，允许同时存在 ARINC429、串口、总线控制器（BC）、远程终端（RT）和总线监控器（BM）操作（工作模式）。但是，只允许用户在同一时刻操作一个窗口，这个窗口的标题栏会变为激活状态，如上图中的“BC 总线控制器”窗口就为激活状态，称这类窗口为活动窗口。

### 4.4.1 板卡号选择

界面可参考图 I-2，当用户向同一设备中插入多该板卡后，用户可以通过对该板卡号的选择来查找自己想要操作的板卡。板卡号范围为 0 到 255。板卡号与对应板卡的顺序选择关系由如下规则决定：

默认距离设备主板上 CPU 最近的为 0 号，最远的为 N 号，255≥N>0。即自距离 CPU 插槽由近到远的顺序，板卡号便会由小变到大。

当用户完成板卡号的选择后，执行“确定”命令便可提交设置给应用程序，程序会自动通过该信息查找板卡，如果找到板卡软件会进入到 MPApp 应用程序的主窗口，等待用户执行新的操作。执行“关闭”或“取消”命令，程序会立刻退出。

### 4.4.2 程序运行模式说明

程序运行时有两种模式，一种是“演示模式”，另一种是“简单应用模式”。当应用程序在提交了板卡号设置后，没有找到指定板卡时，会提示错误信息，如下图所示：

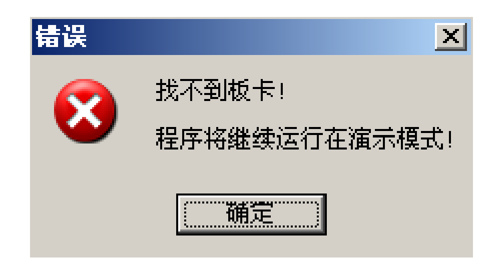



图 S-4.4.1

建议用户在出现该提示后，及时退出应用程序，确认设备中插入了我公司的与应用程序配套的多协议板卡，且板卡的驱动程序被正确安装。

此时程序不会立刻退出，会运行在“演示模式”，在该模式下，程序只会提供基本设置项的演示，不提供数据收发功能的演示。

若程序在提交板卡号后没有出现该错误提示，直接进入到主窗口中，则程序工作在“简单应用模式”下，在该模式下，程序可对板卡提供的所有功能进行操作与简单应用，也可以进行数据通讯操作。

以下的所有说明，主要针对“简单应用模式”进行介绍。

## 4.4.3 1553 部分

### 4.4.3.1 数字量 I/O

数字量I/O窗口

TTL数字量I/O (8路)

数字量输入: 00 (Hex.) 读

数字量输出: 00 (Hex.) 写

开启TTL数字量输出

注: 8位数据代表8路I/O通道.

图 S-4.4.3.1-1

该窗口提供 “8 路 TTL 数字量（输入/输出）”功能。当 MPApp 应用程序工作在“简单应用模式”模式下时，打开窗口会自动从 TTL 端口分别读入一个值，并显示在“数字量输入”窗口中，显示进制为十六进制。

TTL 数字量（输入/输出）共有 8 路，“数字量输入”窗口与“数字量输出”窗口的输入与显示格式均为十六进制数。

执行“读”命令，程序会从 TTL 输入端口读取一个数据，并显示到“数字量输入”窗口中。执行“写”命令，程序会将“数字量输出”窗口中的数值写入到 TTL 输出端口。“开启 TTL 数字量输出”功能，“选中”为开启，“不选中”为不开启。

### 4.4.3.2 设置时间标签与应答超时

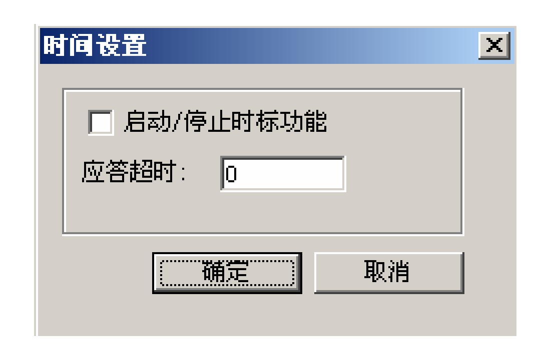



图 S-4.4.3

该窗口提供“启动/停止时标功能”与“应答超时时间设置”两个功能。

| 启动/停止时标功能 | 选中时为“启动时标功能”，未选中时为“停止时标功能”。 |
|---|---|
| 应答超时时间设置 | 应答超时的设置范围为 0 到 65535，单位为 0.5 微秒，实际应答超时时间计算公式如下：<br>实际应答超时时间 = 输入值 * 0.5(微秒) |

#### 4.4.3.3 新建工作模式

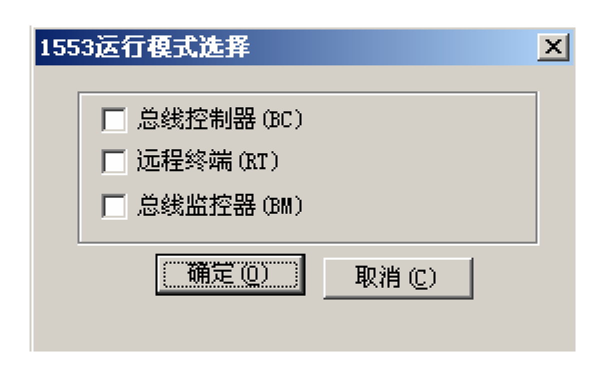



图 S-4.4.3.3-1

1553 协议中有“总线控制器（BC）”、“远程终端（RT）”和“总线监控器（BM）”三个操作（工作模式）。

执行“取消”命令，该窗口不做任何操作；执行“确定”命令，应用程序会创建选中的有效工作模式，当没有任何有效的工作模式被选中时，不做任何操作。

## 4.4.3.4 总线控制器（BC）操作

**1、 设置 BC**

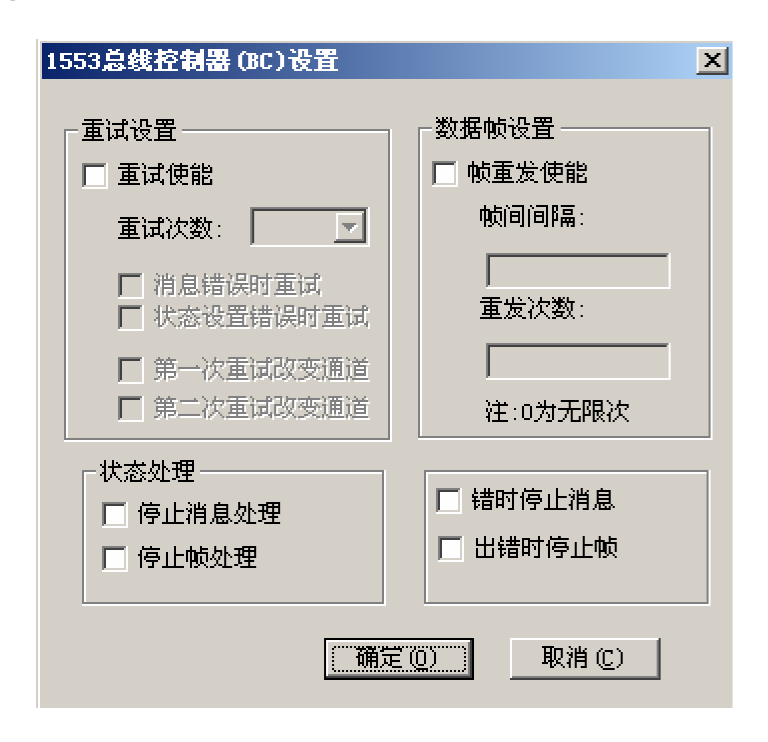



图 S-4.4.3.4-1

执行“确定”命令，程序会将设置提交给硬件，执行“取消”命令关闭该窗口。窗口中的复选项当为“选中”状态时为有效，当为“不选中”状态时为无效。

“数据帧设置”块与“自动重发数据帧”功能相关，在数据通讯时它们之间的关系如下表所示：

| 自动重发数据帧状态（开启/关闭） | 数据帧间时间间隔 | 数据帧自动重发次数 |
|---|---|---|
| 开启 | 有效 | 有效 |
| 关闭 | 有效 | 无效 |

**2、 配置消息**

消息的配置是 1553 数据通讯中必不可少的环节，关于消息的类型、格式及其详细说明，请参考 1553 相关的标准手册。

## 1) 消息的配置与显示




图 S-4.4.3.4-2

上图中所示的窗口为总线控制器（BC）的“消息配置窗口”，用户可通过操作该窗口的功能，实现消息配置。窗口中的列表提供了对已配置消息的显示，程序要求用户一次最多只能配置 4096 条消息。

“消息数量”窗口显示的是当前列表中，已经配置了的消息的总数量。

对消息的操作包括如下操作：

| 操　作 | 说　明 |
| --- | --- |
| 添加消息 | 执行“添加消息”命令进行消息消息添加，程序会按顺序弹出相关的对话窗口，提示用户进行消息的创建与编辑操作。<br>如果选中了“从文件中添加消息”，则程序会提示用户指定一个之前已经保存了的消息文件（.msg），确认后，程序会将文件中的消息读入到窗口的列表中，从文件中读取的消息在添加到列表中时，会遵循如下规则：<br>a. 从现有消息的最后一条消息开始添加；<br>b. 列表中不允许出现同名的消息，在添加时当消息重名，程序提示 |

| | |
|---|---|
| | 是否更新同名消息，用户此时可做出选择；<br>c. 文件中的消息总量数量加上现有列表窗口中的消息总量，如果超过了程序允许的最多添加的消息数量（4096 条），则文件中排列靠后的消息可能不会被添加到列表窗口中。<br>添加消息需要“创建消息窗口”与“消息编辑窗口”组合来使用，才会使消息的添加与编辑变得简便，“创建消息窗口”与“消息编辑窗口”的使用请参考 4.4.6.2.2 和 4.4.6.2.3 的内容。<br>当消息列表中没有消息时，则用户可通过用鼠标双击消息列表的空白处，也可以实现添加消息操作。 |
| 删除消息 | 执行“删除消息”命令可将当前选中位置的消息从列表中删除，如果当前没有被选中的消息位置，则程序会自动从最后一条消息开始删除。 |
| 全部清除 | 执行“全部清除”命令，程序会将列表窗口中的所有消息全部清除掉。 |
| 保　存 | 执行“保存”命令，程序会将列表窗口中现有的消息保存到用户指定的消息文件（.msg）中。 |
| 编辑消息 | 用户可在消息列表中所选的消息上，用鼠标进行双击操作，程序会弹出“消息编辑窗口”供用户对该消息进行编辑。 |

消息列表会将消息内容分段显示，也就是将消息信息进行分类显示，列表提供了如下字段进行消息信息的显示：

| 字段名称 | 说明 |
|---|---|
| 消息名称 | 显示用户为该条消息定义的字称。 |
| 指令字 | 消息中的指令字。 |
| 通道 A/B | 当前消息通讯时选择的通道，A 或 B。 |
| 消息间时间间隔 | 即该消息发送完成后进行延时的时间。单位为 1 微秒。 |
| 允许重试 | 当消息出错时，是否允许消息重试。 |
| 数据 | 显示消息缓冲区中的数据，包括命令字与数据部分，第一个字是指令 |

|  | 字，之后的均为数据，当消息为 RT 到 RT 类型时，则前两个字为指令字，之后的均为数据。 |
|---|---|

执行“确定”操作，程序会将当前消息列表窗口中的消息全部加入消息配置列表，等待下次弹出该窗口时，这些加入消息配置列表的消息，会再次显示在消息列表窗口中。执行“取消”或“关闭”操作，程序不会对消息的配置进行任何修改，当再次打开消息配置窗口时，消息列表中显示的是上次配置的消息。

**2) 创建消息**




图 S-4.4.3.4-3

进行添加操作时，程序最先弹出的是“创建消息窗口”，如上图所示。该窗口提供了对 1553 消息的分类，用户可以通过该分类，先确定创建消息的类型。通过对“总线控制器消息类型选择”和“总线控制器方式代码选择”的操作进行消息分类。

在该窗口中，无论是执行“确定”、“取消”或是“关闭”，都会进入到“消息编辑窗口”，当执行“确定”命令后，消息编辑窗口中会将用户所选的消息类型显示在编辑面板上，以便用户进行消息的详细编辑。执行“取消”或“关闭”命令，程序会将一个默认类型的消息显示在消息编辑窗口中的编辑面板上。这个默认的消息类型为“总线控制器消息类型选择”的第一个类型。

### 3) 编辑消息

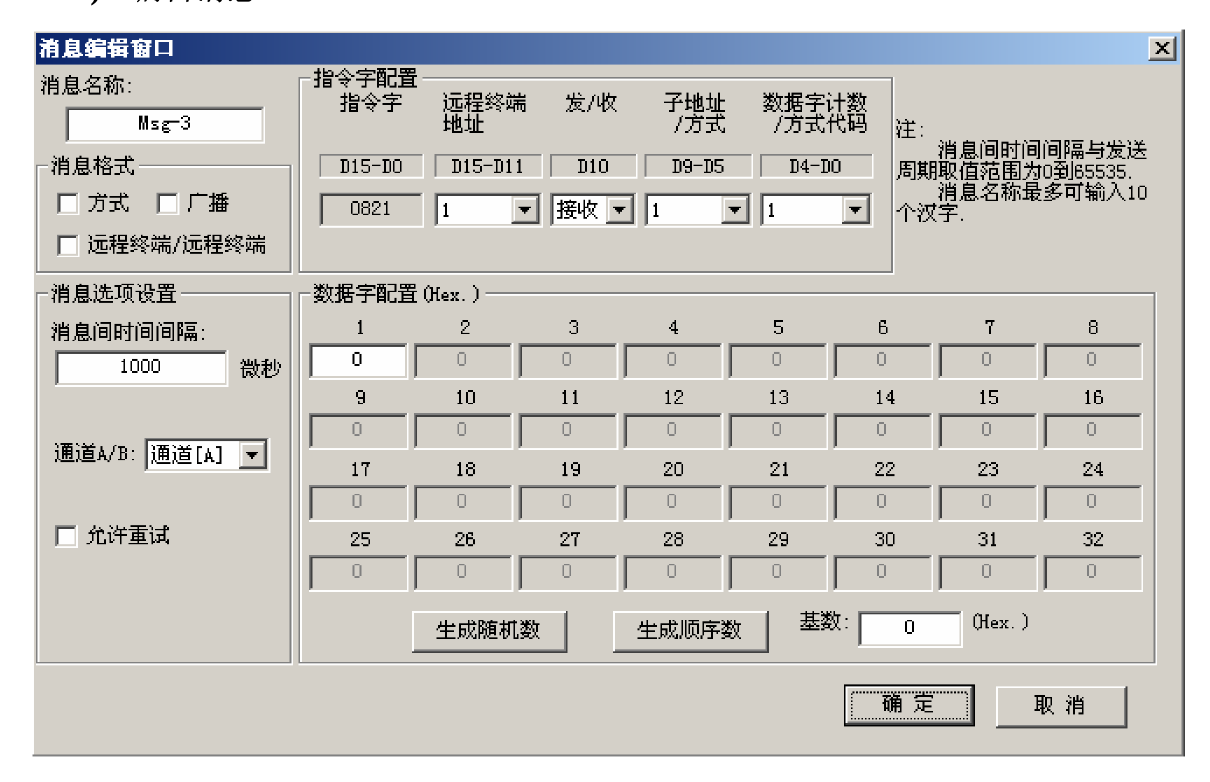



图 S-4.4.3.4-4

当用户执行“编辑消息”或“添加消息”操作时，程序会弹出该窗口让用户对所选消息进行详细编辑。

该面板提供了 1553 消息配置的所有能力，软件还提供了“生成随机数”和“生成顺序数”的能力，使数据配置变得更为简单。执行“生成随机数”命令，程序会按照消息的信息，在“数据字配置区”生成随机数据。执行“生成顺序数”命令前，需要用户指定一个基数值，该命令在生成数据时，会按基数进行累加填入数据。

配置消息时，“消息格式”、“指令字配置”与“数据字配置”是三个相关的操作，改变其中的一个区，另外两个区会做相应的变化，最终的指令字会以十六进制的形式显示在“指令字配置”区的“指令字”项中。

“远程终端/远程终端”当选中时，程序会在“指令字配置”区显示出第二个指令字的配置，如下图所示：




图 S-4.4.3.4-5

执行“确定”命令，程序会将消息编辑面板上的消息信息配置到消息队列中，并会关闭该窗口，最后配置消息窗口中会显示该消息。

如果用户执行的是“添加消息”命令，则程序在确定关闭时，会检查现有消息队列中是否有同名的消息，如果有同名消息存在，程序会提示是否更改消息，如果没有则会将该条新消息添加到“配置消息窗口”中消息列表的最末尾处。

如果用户执行的是“编辑消息”操作，该窗口会显示用户所选消息的内容，用户可在原有内容上进行修改，确认后，程序也会提示是否将原有消息更新。如果用户在编辑消息时，更改了消息的名称，且该消息名称在消息队列中不存在，则程序会对现有的消息进行“添加”操作，窗口关闭后，消息便会出现在“配置消息窗口”中消息列表的最末尾。

用户执行“取消”或“关闭”命令，程序不会对消息列表做任何改动。

**3、 配置消息帧**

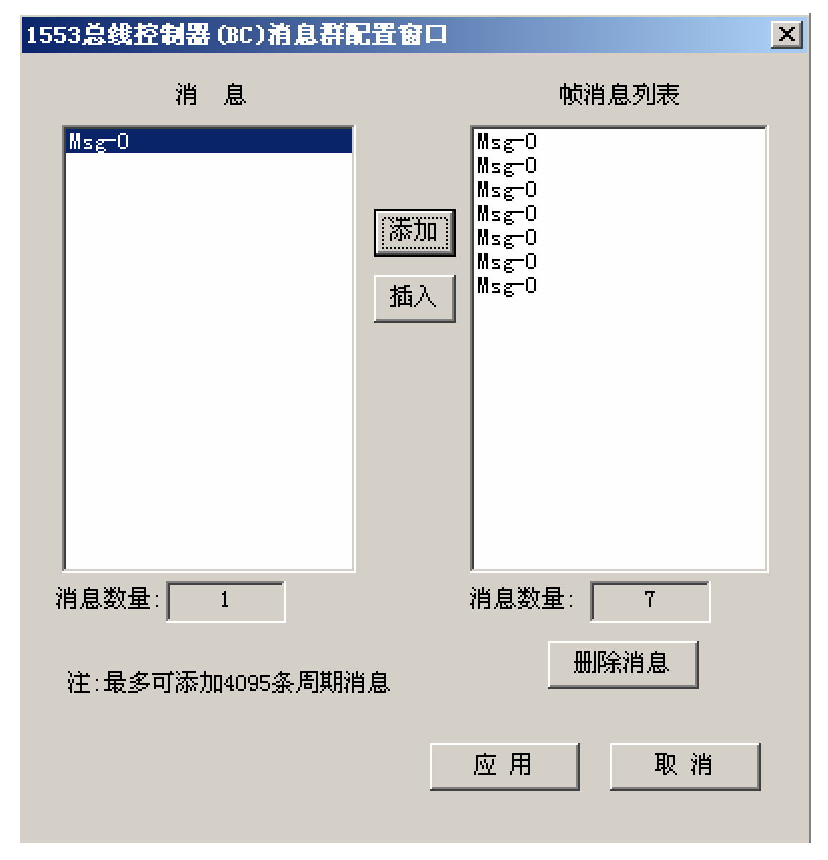



图 S-4.4.3.4-6

消息配置完成后，需要用户将消息组成消息帧，数据在消息通讯时，在总线控制器操作中，发送消息要求打包成消息帧的形式进行传输。上图为总线控制器“帧配置窗口”。

要求用户先配置好至少大于等于 1 条消息的情况下，才能操作该窗口。窗口左边的“消息列表”为用户在“配置消息窗口”中配置好的消息，右边的“帧列表”为用户将要把包到帧中的消息。用户需要从“消息列表”中选择一条消息，然后将其添加到“帧列表”中，用户也可以将“帧列表”中的消息任意地删除，用户打包到帧中的消息数量最多为 4096 条。

插入消息时，需要用户先从“消息列表”中选择一条消息，然后执行“插入”操作，当“帧列表”中没有选择上任何消息时，程序会将“消息列表”中选择的消息插入到“帧列表”中的最前面，如果用户在“帧列表”窗口中选择了一条消息，则程序会将“消息列表”中选择的消息插入到“帧列表”窗口中选择的那条消息的

前面（即前一位置）。

在“消息列表”窗口中，用户用鼠标双击任意消息，那条消息会自动地添加到“帧列表”中，在“帧列表”窗口中，用户用鼠标双击任意消息，那条消息会被自动地删除掉。

执行“应用”命令，可完成消息帧的配置，等下次打开该窗口时，程序会显示出上次配置的结果。执行“取消”或“关闭”命令，程序不做任何操作。

建议：

当用户需要重新改变消息时，建议用户先到“配置消息窗口”将需要修改的消息进行修改，然后再到该窗口中重新将消息帧配置一下（这里指的是执行一下“应用”命令，即再次提交一次帧设置），再进行后面的数据通讯操作。

### 4、 开启/停止接收消息

总线控制器(BC)

M F Rx II Tx Clr

| 序号 | 消息状态字 | 时间标签(us) | 发送方 | 接收方 | 指令字 | 指令字[2] | CNT/MC | 通道A/B | 命令字/数据 | 备注 |
|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 8010 | 66324421 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 7 | 8010 | 66323421 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 6 | 8010 | 66322421 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 5 | 8010 | 66321421 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 4 | 8010 | 64008360 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 3 | 8010 | 64007360 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 2 | 8010 | 64006360 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 1 | 8010 | 64005360 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |

图 S-4.4.3.4-7

用户可通过执行菜单中或工具栏里的“开启接收”与“停止接收”命令，来控制程序对消息的接收显示操作。

当开启接收后，程序会及时地将硬件 BC 缓冲区中的消息数据读取出来，并将消息信息分类显示在接收窗口中。当停止接收后，硬件 BC 也会自动地将消息读入到缓冲区中，但程序不会将消息读出并显示在窗口中。只有再次开启接收后，程序会将 BC 缓冲区中的存放的之前没有由程序接收的旧数据读出并显示到窗口中，再将新数据及时地从缓冲区读出并显示到数据窗口中。程序中 RT 与 BM 的模块对缓冲区的操作与上面描述的相同。

建议用户在进行数据通讯操作前，先将接收开启。

### 5、 发送数据帧

用户可从工具栏与菜单中找到该命令，但要求在发送数据帧前，请确认已经对数据帧进行了配置操作。

**6、 开启/停止自动重发数据帧**

通过执行该命令可开启与停止“自动重发数据帧”功能。工具栏中的按钮被按下或菜单中的命令选中时，表明“自动重发数据帧”为开启状态，否则为“停止状态”。

**7、 清空接收数据**

该操作会将接收数据显示窗口中的所有数据全部清除掉，该操作在“开启接收”状态时无效。

**8、 显示数据**

程序将接收到的消息数据分类，以字段形式显示，如下所示：

| 字段名称 | 说　明 |
| --- | --- |
| 序号 | 按程序接收到消息的先后顺序编号，最新的消息会显示到接收窗口的最上方，起始编号为 1。 |
| 消息描述字 | 双击该项可显示当前消息的具体描述信息。（只有在停止接收状态下有效）。 |
| 时间标签 | 显示消息通讯时的相对时间，单位为 1 微秒。 |
| 发送方 | 指消息数据的发送方。 |
| 接收方 | 指消息数据的接收方。 |
| 指令字 | 消息中的指令字。 |
| 指令字（2） | 当消息为 RT 到 RT 的类型时，该位置显示消息中的第二个指令字。 |
| CNT/MC | 显示消息指令字中的“数据量/方式代码”数值。 |
| 通道 A/B | 显示消息通讯时选用的通道。 |
| 数据/状态字 | 数据与状态字的合集，双击该项可显示当前消息的数据字与状态字集。 |
| 备注 | RT 到 RT 消息与消息出错提示，即该栏会显示两种信息，当用户看到消息出错信息时，可以通过查看“消息描述字”获取更多的信息。 |

### 4.4.3.5 远程终端（RT）操作

**1、 设置 RT**

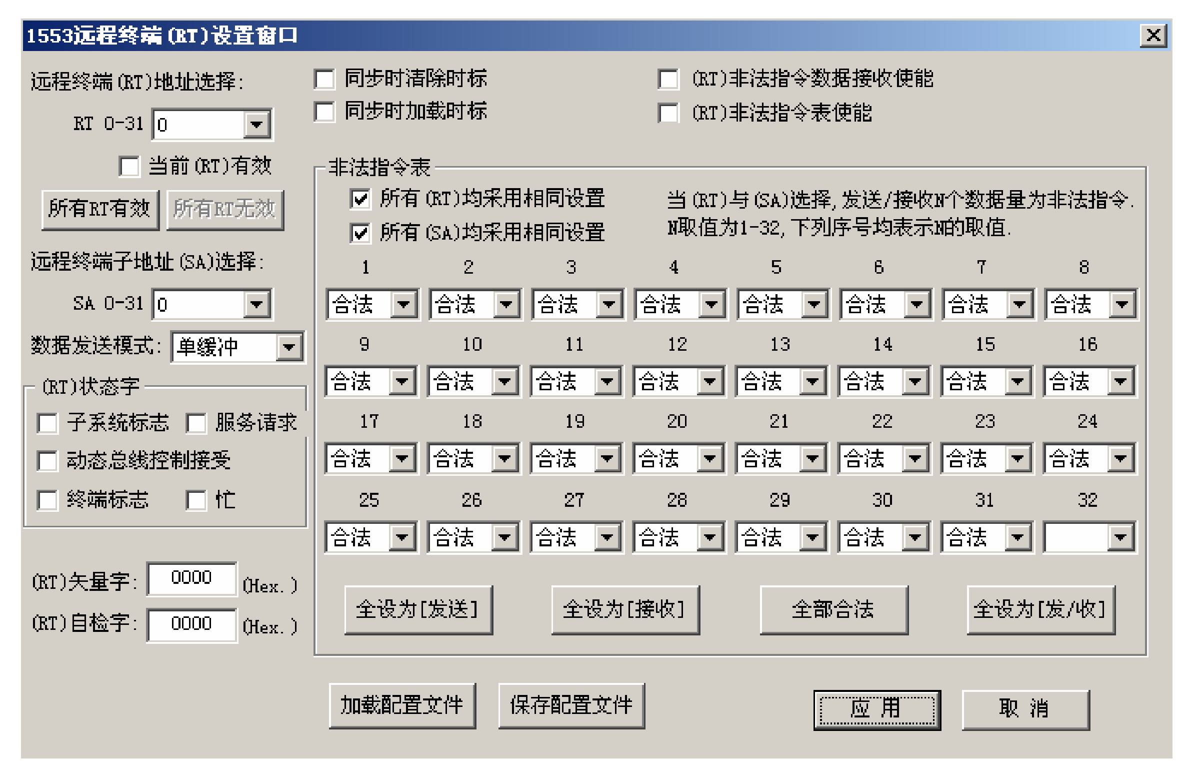



图 S-4.4.3.5-1

该窗口主要是用于对远程终端（RT）与其子地址（SA）功能特性的配置。RT 与 SA 取值均为 0 到 31，当 RT 为 31 时，则表明为广播式通讯，SA 为 0 或 31 时，则表明为方式代码传输，具体内容请参考 1553 标准手册。

当用户对 RT 与 SA 进行选择时，其相应的功能特性项会进行自动保存与显示。

当用户需要 31 个 RT 中任意个有效，参与数据通讯，可以将 RT 地址下的“当前（RT）有效）选中，否则为不参与通讯。

非法指令表在默认情况下是不使能的，用户也可以设置自己的非法指令表，来对 RT 的数据通讯工作进行控制，具体信息请参考“驱动程序使用说明”部分。这里的选择项针对的是数据量合法性。如果用户要让不同的 RT 与 SA 有不同的非法指令表设置，需要先将“所有（RT）均采用桢设置”与“所有（SA）均采用相同设置”置为“不选中”状态。如果这两项被选中，则程序会立即将所有 RT 或 SA 非法指令表项均设为相同设置，用户在更改其中一组 RT 与 SA 非法指令表设置时，也会将所

有的 RT 与 SA 非法指令表设为相同设置。

执行“应用”命令，程序会提交设置给硬件，并在软件中保存本次设置信息，以供下次开启该窗口时参考，如果执行“取消”或“关闭”命令，程序不会做任何操作。

**2、 配置数据**




图 S-4.4.3.5-2

用户可通过操作“数据配置窗口”对远程终端的数据进行配置，配置的内容主要有数据量与数据内容。RT 提供 0 到 31 的地址，每个 RT 下都会有 1 到 30 的 SA 地址。

用户在选择 RT 与 SA 的同时，程序会自动地将设置保存到软件，也会将新的配置信息及时地显示在窗口中。

建议用户在操作时，先选择好 RT 与 SA 的地址，然后对数据量进行选择，数据量为 0 时，表明该 RT 下的 SA 没有数据，数据字最多可有 32 个。活用“生成随机数”与“生成顺序数”会为数据输入提供便利。

执行“应用”命令，程序会提交设置数据给硬件，执行“取消”或“关闭”命

令，程序不会做任何操作。

**3、 开启/停止接收消息**

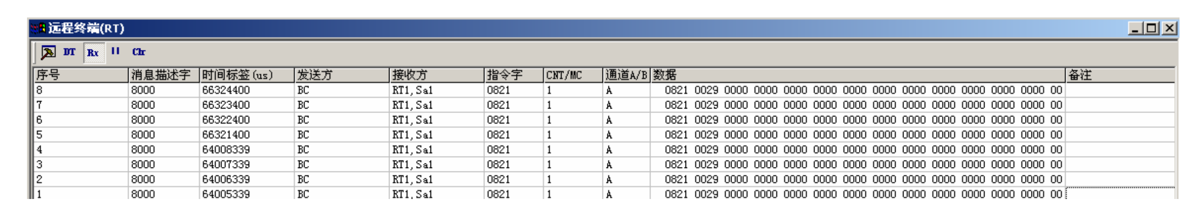



图 S-4.4.3.5-3

操作与 4.4.3.4 相似。

**4、 清空接收数据**

该操作会将接收数据显示窗口中的所有数据全部清除掉，该操作在“开启接收”状态时无效。

**5、 显示数据**

程序将接收到的消息数据分类，以字段形式显示，如下所示：

| 字段名称 | 说　明 |
| --- | --- |
| 序号 | 按程序接收到消息的先后顺序编号，最新的消息会显示到接收窗口的最上方，起始编号为 1。 |
| 消息描述字 | 双击该项可显示当前消息的具体描述信息。（只有在停止接收状态下有效）。 |
| 时间标签 | 显示消息通讯时的相对时间，单位为 1 微秒。 |
| 发送方 | 指消息数据的发送方。 |
| 接收方 | 指消息数据的接收方。 |
| 指令字 | 消息中的指令字。 |
| CNT/MC | 显示消息指令字中的“数据量/方式代码”数值。 |
| 通道 A/B | 显示消息通讯时选用的通道。 |
| 数据 | 数据的合集，双击该项可显示当前消息的数据字集。 |
| 备注 | RT 到 RT 消息与消息出错提示，即该栏会显示两种信息，当用户看到消息出错信息时，可以通过查看“消息描述字”获取更多的信息。 |

### 4.4.3.6 总线监控器（BM）操作

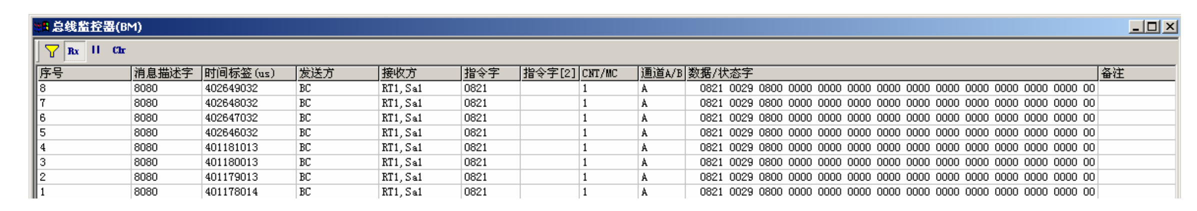
**1、 设置过滤器**




图 S-4.4.3.6-1

提供总线监控器（BM）方式下的监视过滤信息，BM 只会监视设置了过滤信息类型的消息，过滤时依据的条件主要有 RT 地址与“发/收”操作，更多的信息请参考驱动程序使用说明部分。

**2、 开启/停止数据接收消息**




| 序号 | 消息描述字 | 时间标签(us) | 发送方 | 接收方 | 指令字 | 指令字[2] | CNT/MC | 通道A/B | 数据/状态字 | 备注 |
|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 8080 | 402649032 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 7 | 8080 | 402648032 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 6 | 8080 | 402647032 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 5 | 8080 | 402646032 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 4 | 8080 | 401181013 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 3 | 8080 | 401180013 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 2 | 8080 | 401179013 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |
| 1 | 8080 | 401178014 | BC | RT1,Sa1 | 0821 | | 1 | A | 0821 0029 0800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00 | |

图 S-4.4.3.6-2

操作与 4.4.3.4 相似。

**3、 清空接收数据**

该操作会将接收数据显示窗口中的所有数据全部清除掉，该操作在“开启接收”状态时无效。

**4、 显示数据**

程序会将及时接收到的消息数据信息分类显示。

## 4.4.4 ARINC429 部分

### 4.4.4.1 ARINC429 通讯主窗口




图 S-4.4.4.1-1

**1、 数据的接收**

需要进行数据接收时，执行“开始接收”就可以了，执行它后，该按钮会自动变为“停止接收”按钮，终止接收操作时，只需再次点击一下该按钮就可以了。

程序会将接收到的数据分别显示在列表窗口中，接收到的数据量会显示在列表窗口下方的文本窗口中。查看接收数据可以通过选择接收通道卡实现。

接收数据时，如果添加了时标功能，在接收列表中的数据显示会有所变化。在每一个数据后面都会加上“时:分:秒.毫秒”的显示，用来表示当前数据接收到的时间。如下图所示：

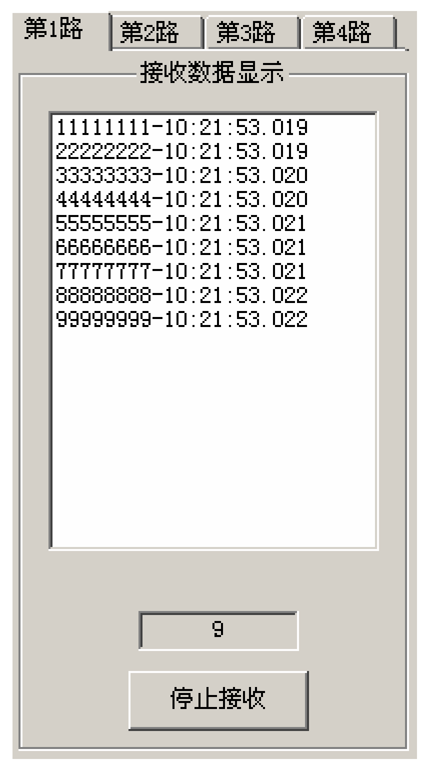



图 S-4.4.4.1-2

**2、 数据的发送**

发送数据时可以通过执行“发送数据”进行数据发送，如果配置了定时发送功能，则可通过执行“停止定时发送”停止定时发送。

程序总是会将发送列表中的数据发送出去，所示要保证发送数据列表不为空。添加发送数据到列表有两个方法，一个就是通过“添加”、“删除”、“清空”操作将数据文本窗口中的数据添加到列表，输入数据时要以十六进制的输入方式进行输入，列表中显示的数据也都是十六进制的。另一个方法是通过“加载数据文件”和“保存数据文件”进行发送列表的配置，“加载数据文件”会指定一个数据文件，并将里面的数据以续加的方式添加到列表中，不会将列表中原有的数据清空。“保存数据文件”会将列表中的数据保存到文件中。两种配置发送数据列表的方法可以混合使用。

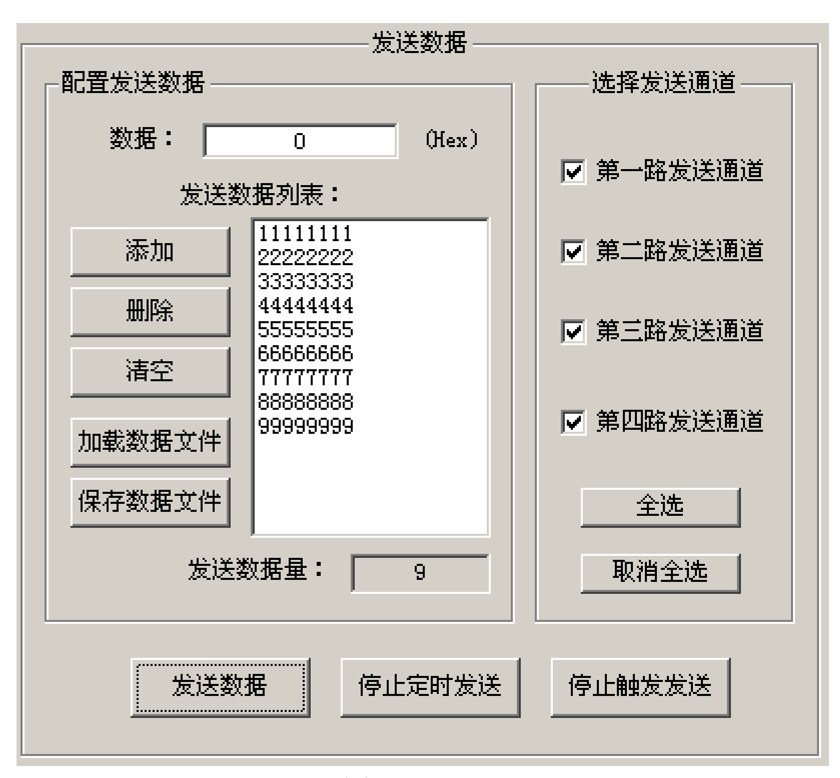



图 S-4.4.4.1-3

注：“删除”操作会将发送列表中被选中的数据删除掉，当列表中没有数据被选中时，将会从列表中的最后一个数据开始删除。

### 4.4.4.2 ARINC429 设置窗口

图 S-4.4.4.2-1

## 1、 设置配置字

配置字包括：比特率、数据位和校验，如下表所示：

| 配置字内容 | 说明 | | |
|---|---|---|---|
| 比特率 | 100K | 48K | 12.5K |
| 校验 | 奇校验 | 偶校验 | 无 |

第 1 路发送与第 1、2 路接收通道均采用相同的配置字，在设置窗口中共有 4 个配置卡，按这种规则，配置字对发送与接收通道的影响关系如下表所示：

| 发送通道 | 影响关系 |
|---|---|
| 第 1 路发送 | 第 1、2 路接收与第 1 路发送 |

| 第 2 路发送 | 第 2 路发送 |
|---|---|
| 第 3 路发送 | 第 3、4 路接收与第 3 路发送 |
| 第 4 路发送 | 第 4 路发送 |

修改任意一个配置卡中的配置字，与其存在相互影响关系的另一个配置卡中的配置字也会随之改变，软件会自动确保配置字在设置时保持一致。

**2、 设置触发深度**

| 缓冲区名称 | 触发深度取值范围 |
|---|---|
| 发送缓冲区 | 0 到 254 |
| 接收缓冲区 | 0 到 510 |

触发深度的设置，会直接影响到 FIFO 缓冲区状态，如果设置值为 0，表示不采用触发深度功能，如果设置值大于 0，则表示采用触发深度。

例如在接收时，需要定时接收定长的数据，可以将数据长度设置好，设置值为待接收长度-1。例如，当需要一次接收至少 6 个数据时，则触发深度可设置为 5，待程序判断缓冲区状态为到达触发深度时，便可将定长的数据一并进行接收。需要注意的是，如果用户设置了时间标签功能，则时间标签也会被算为数据中的一个长度。

建议用户在使用该深度时最好不要超过整体深度设置值的三分之二。

**3、 设置定时发送**

定时发送是指将缓冲区内的数据按指定的时间间隔发送出去。本程序提供了对四路发送通道的定时发送功能的设置。

定时时间的基本单位为 50 微秒，当输入值为 20000 时，则表示实际定时发送的时间间隔为 1 秒。具体公式如下：

实际时间间隔 = 时间间隔输入值 × 单位时间

经过如上公式，我们可以算出实际的时间间隔值：

实际时间间隔 = 20000 × 50

= 1000000（微秒）= 1（秒）

定时发送值设置为 0 时，则表示不采用定时发送功能。

**4、 添加时间标签**

时间标签是指在接收到的数据后面增加一个接收到的时间，这个时间就称为时间标签，该标签的作用是用来描述当前接收到的这个数据的到达时间。其单位是 50 微秒，求得实际到达时间的方法与计算定时发送时间间隔相同，但这个时间标签的定时是从设置该功能的时候开始的，它的计时完全由硬件来实现。

**5、 设置接收标号过滤**

标号过滤实际上就是，通知硬件只接收那些用户想收到的数据。该功能的实现主要是依据 ARINC429 标准，通过识别数据的标号实现数据过滤功能，标号指的是 ARINC429 数据中的第 0 位到第 7 位表求的一个 8 位二进制数据。

当数据位设置为 32 位时，SD 值也参与过滤，SD 值指 ARINC429 数据中的第 11、12 两位，SD 下拉列表中的数据为 0 到 3，描述二进制数据的两个数据位。

当数据位设置为 25 位时，SD 值则不参与过滤，软件中每一个 SD 值都对应着 256 个标号。设置标号过滤时，与 32 位的有些区别，需要将 4 个 SD 值对应的 4 组标号均设置为相同的，而在 32 位数据通讯时，由于 SD 参与了过滤，则每个 SD 值都可以对应不同的标号设置 S。

在接收标号过滤设置中，共有 256 个标号可设置，分别是从 0 到 255。且每一个 SD 值都有 256 个标号。标号里横向的是低 4 位，纵向的是高 4 位，如果用户要设置一个 SD 为 2 标号为 45H（69D）的数据过滤时，需要先将 SD 选择在 2 上，然后再在相应的标号上选中即可，操作结果如下所示：

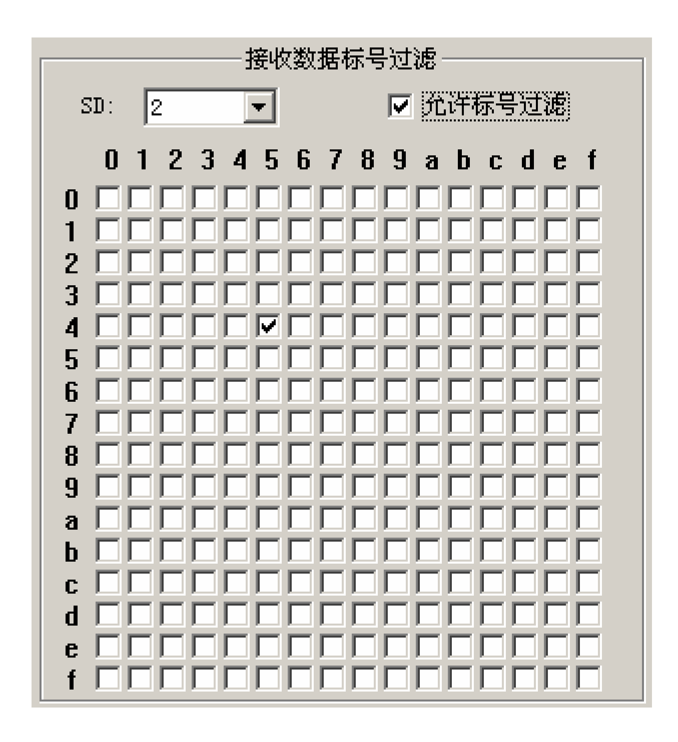



图 S-4.4.4.2-2

在设置 256 个标号时当用户选中了其中的某一项,就表示对该标号的数据进行过滤，也就是会接收到该标号的数据。

单配置了标号与 SD 是不够的，只有将“允许标号过滤”选中，才可以便板卡具有标号过滤功能。

6、 板卡工作方式

| 方式 | 说明 |
| --- | --- |
| 自环 | 在该方式下，板卡 ARINC429 功能实现内部自环功能，主要表现为，板卡可以在不与外界通讯时，便可接收到自己发送的数据，板卡具有 4 路接收与 4 路发送的能力，通过本程序可以从 4 路接收通道接收到发送的数据，但需要注意的是，此时奇数序列接收通道收到的是原数据，偶数序列接收通道接收到的数据为原数据的取反值。该工作方式下也存在和配置字设置同样的对应关系，例如，从第 1 路发送的数据会由 1、2 路接收收到，第 3 路发送的数据会由 3、4 路接收收到。 |

| 正常 | 在该方式下，板卡可保证正常通讯。 |
|---|---|

7、 配置的保存与加载

在每次应用程序打开后，程序会自动按照默认值对板卡进行配置，实现基本的数据通讯能力，但往往用户都需要将板卡的功能配置成自己想要的功能，这样就需要经常对板卡进行配置操作，在设置窗口中的“加载配置”与“保存配置”可以提供些方便。当用户进入到配置窗口后，首次对板卡通讯进行自定义配置，配置完成后，可以执行“保存配置”，这时程序会将程序中当前的配置保存到配置文件中，以便下次重新启动应用程序后，用户可以通过执行“加载配置”操作重新获取之前保存的配置。

## 4.4.5 串口部分

### 4.4.5.1 串口通讯主窗口



图 S-4.4.5.1-1

**1、 数据的接收**

需要进行数据接收时，执行“开始接收”就可以了，执行它后，该按钮会自动变为“停止接收”按钮，终止接收操作时，只需再次点击一下该按钮就可以了。程序会将接收到的数据分别显示在列表窗口中，接收到的数据量会显示在列表窗口下方的文本窗口中，如下图所示。查看接收数据可以通过选择接收通道卡实现。

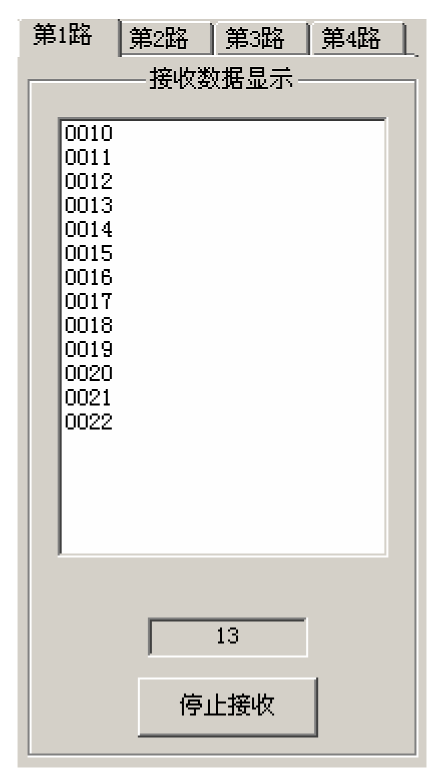



图 S-4.4.5.1-2

**2、 数据的发送**

发送数据时可以通过执行“发送数据”进行数据发送。程序总是会将发送列表中的数据发送出去，所示要保证发送数据列表不为空。添加发送数据到列表有两个方法，一个就是通过“添加”、“删除”、“清空”操作将数据文本窗口中的数据添加到列表，输入数据时要以十六进制的输入方式进行输入，列表中显示的数据也都是十六进制的。另一个方法是通过“加载数据文件”和“保存数据文件”进行发送列表的配置，“加载数据文件”会指定一个数据文件，并将里

面的数据以续加的方式添加到列表中，不会将列表中原有的数据清空。“保存数据文件”会将列表中的数据保存到文件中。两种配置发送数据列表的方法可以混合使用。




图 S-4.4.5.1-3

注：“删除”操作会将发送列表中被选中的数据删除掉，当列表中没有数据被选中时，将会从列表中的最后一个数据开始删除。

### 4.4.5.2 串口设置窗口

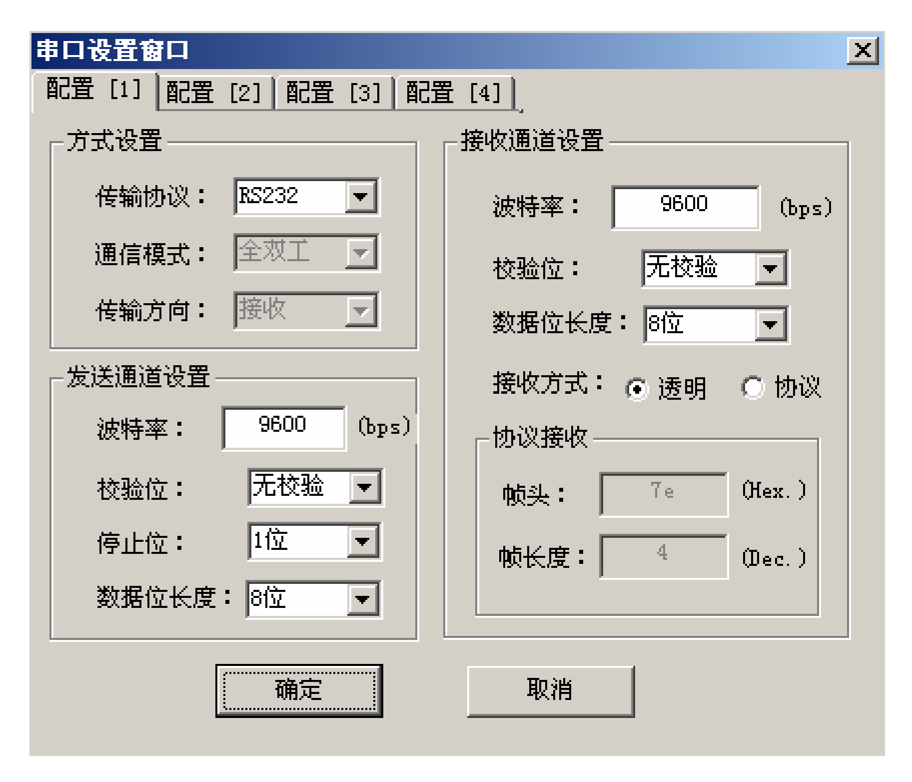



图 S-4.4.5.2-1

## 1、 设置工作方式

当传输协议为 RS422/485 时，需要设置通信模式为全双工或半双工，若通信模式为半双工时，还要选择传输方向为发送或接收。若传输模式为 RS232 时其他两项不可用

## 2、 设置发送通道配置

配置包括：波特率、校验位、停止位和数据位长度，如下表所示：

| 配置字内容 | 说明 | | | |
|---|---|---|---|---|
| 波特率 | 用户自行输入，输入的值为能被 12M 整除的正整数 | | | |
| 校验位 | 奇校验 | 偶校验 | 无 | |
| 停止位 | 1 位 | | 2 位 | |
| 数据位长度 | 8 位 | 7 位 | 6 位 | 5 位 |

## 3、 设置接收通道配置

配置包括：波特率、校验位、数据位长度和接收方式，如下表所示

| 配置字内容 | 说明 |
|---|---|

| 波特率 | 用户自行输入，输入的值为能被 12M 整除的正整数 | | | |
|---|---|---|---|---|
| 校验位 | 无 | 奇校验 | 偶校验 | |
| 数据位长度 | 8 位 | 7 位 | 6 位 | 5 位 |
| 接收方式 | 透明接收 | | 协议接收 | |

当接收方式选择了协议接收时还需要设置接收帧的帧头和帧长度，当选择了透明接收时帧头和帧长度设置不可用。

## 4.4.6 工具条命令

### 4.4.9.1 MPApp 主窗口工具条命令

工具栏：



功能：

| | | |
|---|---|---|
| | 新建工作模式 | 创建 BC、RT、BM 操作（工作模式）。 |
| | 设置时标与超时 | 显示“启动/停止时标与应答超时设置窗口”。 |
| | 数字量 I/O | 显示数字量 I/O 窗口。 |
| | 帮助主题 | 提供您可从其得到帮助的主题索引。 |

### 4.4.9.2 总线控制器（BC）工具条命令

总线控制器工具栏：



总线控制器工具栏中的操作与总线控制器菜单命令功能相对应。

| | | |
|---|---|---|
| | 设置 | 显示设置窗口。 |
| | 消息 | 显示消息配置窗口。 |
| | 组帧 | 显示帧配置窗口。 |

| Rx | 运行 | 开启接收操作。 |
| --- | --- | --- |
| II | 停止 | 停止接收操作。 |
| Tx | 发送数据帧 | 将用户配置好的数据帧写入硬件。 |
|  | 自动重发数据帧 | 开启/停止 数据帧的“自动重发”功能。 |
| Clr | 清空接收数据 | 清空接收数据显示列表。 |

注：

操作菜单中的“设置”、“消息”、“组帧”与“清空接收数据”时，需要处于停止接收状态。

### 4.4.9.3 远程终端（RT）工具条命令

远程终端工具栏：



远程终端工具栏中的操作与远程终端菜单命令功能相对应。

|  | 设置 | 显示设置窗口。 |
| --- | --- | --- |
| DT | 数据配置 | 显示数据配置窗口。 |
| Rx | 运行 | 开启接收操作。 |
| II | 停止 | 停止接收操作。 |
| Clr | 清空接收数据 | 清空接收数据显示列表。 |

注：

操作菜单中的“设置”、与“清空接收数据”时，需要处于停止接收状态。

### 4.4.9.4 总线监控器（BM）工具条命令

总线监控器工具栏：



总线监控器工具栏中的操作与总线监控器菜单命令功能相对应。

| | | |
|---|---|---|
| | 过滤器 | 显示过滤设置窗口。 |
| Rx | 运行 | 开启接收操作。 |
| II | 停止 | 停止接收操作。 |
| Clr | 清空接收数据 | 清空接收数据显示列表。 |

注：

操作菜单中的“过滤器”、与“清空接收数据”时，需要处于停止接收状态。

### 4.4.7 帮助

软件提供了联机帮助的功能，当用户遇到不知道的操作，或者是不明确窗口或某些命令的功能时，可以用鼠标点中，并按下 F1 键，查看帮助中的说明。用户也可以通过“帮助”菜单中直接打开并查看“帮助”内容。

### 4.4.8 退出程序

建议用户停止并关闭当前所有的工作模式操作，然后执行“1553”菜单中的“退出”命令，或点击主窗口右上角的“关闭按钮”，可以退出程序。

## 4.5 1553 部分数据的打开与保存

如果没有创建任意工作模式，则无法进行“打开”与“保存”数据操作。“打开”与“保存”数据操作只能对当前工作模式下的数据进行操作。

软件共包括 BC、RT、MT 三种工作模式，每种工作模式都有“打开”与“保存”数据的功能，但数据类型有“通讯数据”与“配置数据”两种。“通讯数据”文件中存放的是工作模式窗口中显示接收到的消息数据，“配置数据”文件中存放的是板卡设置数据。“打开”与“保存”操作的界面也因此分为了两个形式：

一种是用于打开或保存“通讯数据”，用户可从“工作模式”菜单中找到“打开”与“保存”数据命令，也可以从程序主窗口的工具栏里找到“打开”与“保存”命令；

另一种是用于打开或保存“配置数据”，用户可从那些复杂的硬件功能设置窗口

中找到，有效地使用配置数据文件操作，可以提高硬件功能配置的效率，也会减少用户重复地进行复杂的配置工作的次数。

注：当正在执行“打开”或“保存”类的操作时，用户无法立刻终止当前操作，或是进行其它工作，建议用户等待该类操作完成后，再进行下一步操作。